BAB II
TINJAUAN PUSTAKA

2.1 Kajian Pustaka

Penelitian terkait *chatbot* dengan menggunakan transfer learning sendiri sudah banyak dilakukan dan diterapkan secara luas di berbagai bidang kehidupan. Pada penelitian-penelitian sebelumnya yang menggunakan Transfer Learning BERT, dijadikan dasar untuk penelitian sehingga menjadi lebih baik lagi dan dapat membantu penelitian selanjutnya. Berikut penelitian terdahulu menurut penulis relevan dengan penelitian, yang disajikan pada Tabel 2.1.

Tabel 2. 1 Penelitian Terdahulu

No	Peneliti	Objek	*Dataset*	Metode	Hasil
1	SoYeop Yoo, OkRan Jeong (2019) [14]	*Chatbot*	*Dataset* Twitter berisi sekitar 15.000.000 *tweet*. Dataset berita berisi sekitar 102.000 artikel *Dataset TACRED* berisi sekitar 106.000 kalimat	BERT dan *Knowledge Graph*	Hasil pada pengujian *relation extraction* model BERT mendapat performa yang baik dengan nilai *F1 Score* 75,7. Serta saat diaplikasikan dapat menjawab pertanyaan dengan baik.
2	CAIRONE Fiorentino (2021) [15]	*Edu Chatbot*	Dataset mengenai topik-topik kursus secara teoritis dan praktis Bahasa pemrograman *Java* dalam pemrograman	BERT, SpaCy dan ConveRT	Hasil penelitian menunjukan untuk tingkat *END-TO-END* dan *ACTION* metode *ConveRT* dapat menjawab 100% pertanyaan

No	Peneliti	Objek	*Dataset*	Metode	Hasil
			berorientasi objek.		dengan benar dibandingkan *BERT* dan SpaCY yang hanya dapat 97% dan 96% dalam menjawab pertanyaan dengan benar.
3	Kevin Peyton Dan Saritha Unnikrishnan (2023) [16]	*Chatbot*	Data tanya jawab online yang tersedia untuk calon mahasiswa online calon mahasiswa online di Universitas Teknologi	Sentence BERT (SBERT)	Hasilnya menunjukkan bahwa SBERT memberikan hasil yang paling menjanjikan dengan skor *F1* 0,99 pada dataset khusus ini, diikuti oleh *Dialogflow* dengan skor *F1* 0,96 dan *QnA Maker* dengan skor F1 0,95.
4	Nguyen Thi Mai Trang dan Maxim Shcherbakov (2020) [13]	*Vietnamese Chatbot*	*Dataset* dari Wikipedia bahasa *Vietnam* dataset dari kompetisi *online Zalo*	M-BERT, DmBERT, mBERT_XQuAD *dan* monolingual PhoBERT	Model BERT khusus Vietnam *(*PhoBERTbase*)* mendapatkan mendapatkan skor *F1* terbaik pada set validasi. Model mBERT_XQuAD mendapatkan skor *F1* tertinggi dalam model multibahasa (M-BERT, DmBERT, mBERT_XQuA*D* dengan skor masing-masing

No	Peneliti	Objek	*Dataset*	Metode	Hasil
					0,792, 0,780, 0,802 secara berurutan)
5	Marvin Chandra Wijaya (2021) [17]	*Automatic Short Answer Grading System*	Kumpulan data terdiri dari jawaban singkat yang dibuat untuk kuis antara siswa dari sekolah menengah di Bandung, Indonesia.	BERT	Hasil pengukuran output BERT dari sistem yang diimplementasikan memiliki nilai koefisien *Cohen Kappa* sebesar 0.75, dan dengan *confusion matrix* mendapat nilai *Precision* sebesar 0.94, *Recall* sebesar 0.96, *Specificity* sebesar 0.76, dan *F1 Score* sebesar 0.95.
6	Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, dan Jimmy Lin (2019) [12]	*End-o-End Question Answer System*	Data Pertanyaan tes berasal dari set pengembangan *SQuAD*	*Anserini Retriever* + BERTSerini	Hasilnya adanya peningkatan besar dari hasil sebelumnya pada koleksi tes standar tolak ukur, yang menunjukkan bahwa penyempurnaan yang telah dilatih sebelumnya dengan *SQuAD* cukup untuk mencapai akurasi tinggi dalam mengidentifikasi rentang jawaban
7	Muhammad Rana (2019) [18]	*Question Answering* di specific *domain*	Pertanyaan Umum dari beberapa halaman web	BERT	Menurut hasil, untuk hasil ekstraksi jawaban dari

No	Peneliti	Objek	*Dataset*	Metode	Hasil
			FAQ Georgia Southern		dokumen besar BERT-Base mendapat akurasi yang tertinggi. Dan dengan BERT-base juga mendapat akurasi pada ekstraksi jawaban dari dokumen yang dipecah - pecah.
8	Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, dan Mohit Iyyer (2019) [19]	*Conversational Question Answering*	QuAC *(Question Answering in Context)* berisi dialog interaktif antara pencari informasi dan penyedia informasi.	BiDAF + FlowQ + BERT	Hasil menunjukkan menggabungkan riwayat percakapan menggunakan *BERT* secara signifikan meningkatkan kinerja dalam *Conversational Question Answering* (ConvQA), dan metode pemodelan riwayat yang diusulkan efektif dan memiliki efisiensi pelatihan yang lebih baik dibandingkan dengan pendekatan sebelumnya
9	Herman Jansson (2021) [20]	*Multilingual* dan *Swedish monolingua*	Kumpulan data berisi pertanyaan yang dikumpulkan	alBERT + BERT + m-distilBERT + m-BERT + BERT	Hasilnya semua model dapat menjalankan percobaan kecuali model

No	Peneliti	Objek	*Dataset*	Metode	Hasil
		l question answer	dari situs web *CSN*	dan XLM-RoBERTa	alBERT. Dan model yang telah disesuaikan berdasarkan model yang telah dilatih di Swedia dan kumpulan data *SQuADv2 Swedia* lebih unggul dalam semua matrik evaluasi kecuali kecepatan.
10	Mahanti Indah Rahajeng dan Ayu Purwarianti (2021) [21]	*Indonesian QA system*	*Dataset* berupa data *QA* yang berhubungan dengan entitas organisasi, *entitas* produk, dan properti produk.	IndoBERT	Hasilnya *Fine-Tuning IndoBERT* mendapatkan hasil yang lebih baik dibandingkan dengan model lain.
11	Muhammad Rahaji Jhaerol (2023) [22]	*AI Chatbot MBKM*	*QA* seputar MBKM	LSTM	Hasil model cukup baik dengan akurasi 81%, namun tidak dapat menangani pertanyaan diluar konteks pengetahuan *chatbot*.
12	Abonia Sojasingarayar(2020) [2]	AI Chatbot	Data Corpus Cornell Movie Subtitle dan metadata movie lainnya.	Seq2Seq + *Attention Mechanism*	Hasilnya model yang dibuat cukup baik dalam memberikan jawaban dari pertanyaan yang diberikan, namun saat melakukan *training* model

No	Peneliti	Objek	*Dataset*	Metode	Hasil
					membutuhkan waktu yang cukup lama dan perlunya perangkat dengan proses komputasi yang tinggi. Serta sulitnya dalam menemukan *hyperparameter* yang tepat saat melakukan *training*.

Berdasarkan Tabel 2.1, peneliti mendapatkan bahwa pemakaian BERT untuk *chatbot* ataupun *question answer* berbahasa inggris maupun bahasa lain sudah banyak banyak dilakukan baik menggunakan BERT dengan *corpus* inggris ataupun *corpus* lain. Namun untuk pemakaian BERT dalam Bahasa Indonesia masih sedikit dilakukan, dalam penelitian terdahulu yang dikumpulkan hanya ada dua penelitian yang dapat ditemukan pemakaian BERT untuk *question answer* dan IndoBERT untuk *Chatbot*. Pada penelitian lain yang dilakukan Muhammad Rahaji Jhaerol mengenai membuat AI *Chatbot* untuk MBKM menggunakan *Long short-term memory* (LSTM) mendapatkan bahwa membangun chatbot dengan metode *Deep Learning* tetap dapat dilakukan namun hasilnya kurang memuaskan dalam menangani pertanyaan diluar konteks pengetahuan *chatbot* karena keterbatasan datasetnya. Penelitian mengenai IndoBERT di Tabel 2.1 menjadi literatur utama penelitian ini dalam membangunnya, dengan penelitian mengenai Seq2Seq dan LSTM untuk *chatbot* sebagai pendukung dalam pembuatan *chatbot* serta penelitian lain mengenai pemakaian BERT dalam *chatbot* yang ada ditabel sebagai pendukung dalam referensi pembuatan ataupun pengujian evaluasi *chatbot.*

2.2 Landasan Teori

Landasan teori memuat tentang pengetahuan atau informasi yang berkaitan dengan penelitian yang dilakukan untuk membantu peneliti dalam melakukan penelitian. Berikut ini merupakan landasan teori yang berkaitan dengan penelitian ini.

2.2.1 *Chatbot*

Chatbot adalah perangkat lunak berbasis *AI* yang mampu berinteraksi dengan manusia menggunakan bahasa alami. Penggunaan chatbot sangat umum sebagai titik awal dalam memberikan dukungan kepada pelanggan, dan telah terbukti sangat efektif dalam menjawab pertanyaan sederhana pengguna. *Chatbot* dapat berupa model sederhana berdasarkan aturan tertentu, atau model yang sangat canggih, tergantung pada kebutuhan bisnis. Sebagian besar chatbot yang digunakan di industri saat ini dilatih untuk mengarahkan pengguna ke sumber informasi yang relevan atau merespons pertanyaan yang terkait dengan topik tertentu. Sangat tidak mungkin memiliki *chatbot* yang dapat menjawab pertanyaan terkait dengan berbagai bidang secara umum. Hal ini disebabkan karena melatih *chatbot* untuk topik tertentu memerlukan jumlah data yang besar, dan melatih *chatbot* pada banyak topik dapat menyebabkan masalah kinerja [10].

2.2.2 *Deep Learning*

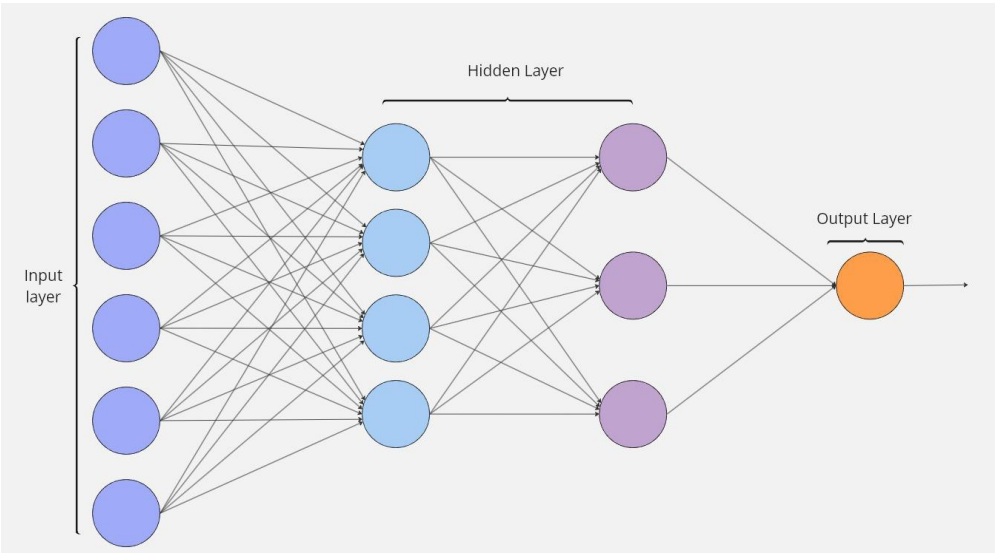
Deep learning adalah subbidang khusus dalam *machine learning* yang memiliki pendekatan baru dalam merepresentasikan pembelajaran dari data, yaitu dengan penekanan pada pembelajaran dari lapisan representasi yang berturut-turut dan semakin bermakna [23]. *Deep learning* adalah cabang dari *machine learning* yang didasarkan pada serangkaian algoritme yang berusaha memodelkan abstraksi tingkat tinggi dalam data.

Perkembangan *deep learning* terjadi sejalan dengan penelitian dalam bidang *artificial intelligence*, khususnya dalam studi *neural network*. Perkembangan ini terutama terjadi pada tahun 1980-an, dan daerah ini berkembang pesat, terutama berkat kolaborasi Geoff Hinton dengan para ahli *machine learning*. Pada saat itu, teknologi komputer belum cukup maju untuk mencapai kemajuan yang signifikan

dalam hal ini, sehingga kita harus menunggu ketersediaan data yang lebih banyak serta daya komputasi yang jauh lebih baik untuk melihat kemajuan yang nyata [24]. Istilah "*deep*" dalam *deep learning* merujuk pada kedalaman arsitektur dari jaringan saraf tiruan, sedangkan "*learning*" merujuk pada pembelajaran yang dilakukan oleh *artificial neural network* itu sendiri.

Meskipun *artificial neural network* dan model dalam *deep learning pada* dasarnya memiliki struktur yang sama, hal ini tidak berarti bahwa kombinasi dari dua *artificial neural network* akan memiliki kinerja yang serupa dengan *artificial neural network* ketika dilatih menggunakan data. *neural network* merupakan paradigma yang terinspirasi dari biologi (meniru cara kerja otak mamalia) yang memungkinkan komputer untuk belajar dari data observasi.

Saat ini, *neural network* memberikan solusi untuk berbagai masalah, *seperti image recognition*, *handwriting recognition*, *speech recognition*, *speech analysis*, dan NLP [25]. Gambar 2.1 mengilustrasikan arsitektur dari *neural network* yang terdiri dari beberapa lapisan, termasuk *input layer*, *hidden layer*, dan *output layer*. *Neural network* ini menggunakan metode *feedforward* (meneruskan informasi dari *input* ke *output* tanpa siklus) dan memiliki bobot (*weights*) yang terhubung di antara setiap *neuron*.



Gambar 2. 1 Arsitektur *Neural Network* [25].

Neural network sendiri memiliki beberapa jenis aristektur berdasarkan arsitektur dan penggunaannya. Berikut jenis arsitektur *neural network* yang sering dipakai.

1. *Feedforward Neural Networks* (FNN)

Feedforward Neural Networks (FNN) adalah unit dasar dari keluarga neural network [25]. FNN bisa diilustrasikan sebagai *unidirectional neural networks* yang tidak memiliki umpan balik atau perulangan dalam strukturnya. Arsitektur FNN mencakup beberapa lapisan tersembunyi dengan sejumlah unit tersembunyi di setiap lapisan. Alasan mengapa jaringan saraf ini disebut *feed-forward* adalah karena tidak ada umpan balik antara lapisan-lapisan tersebut selama operasi normal ketika FNN berfungsi sebagai pengklasifikasi [26].

2. *Convolutional Neural Networks* (CNN)

Convolutional Neural Networks (CNN) merupakan algoritme *deep learning* yang menggunakan gambar sebagai input dan memberikan bobot dan bias pada berbagai aspek dalam gambar tersebut. Proses prapemrosesan pada CNN relatif lebih sederhana dibandingkan dengan algoritme klasifikasi lainnya, karena algoritma ini memiliki kemampuan untuk belajar filter dan karakteristik dari data. Seperti *neural network* lainnya, CNN juga terdiri dari neuron dan memiliki bobot dan bias yang dapat dipelajari. Jumlah tertimbang diambil dari beberapa input yang diterima, dan kemudian dilalui melalui fungsi aktivasi bersamaan dengan output.

Keunikan CNN terletak pada cara kerjanya terhadap volume data. Inputnya bukan vektor, tetapi gambar multisaluran. CNN digunakan dalam pemrosesan gambar karena dapat menangkap ketergantungan spasial dan temporal dalam sebuah gambar dengan baik melalui penerapan filter yang relevan. Dengan begitu, *network* ini mampu memahami gambar lebih baik dan melakukan penyesuaian yang lebih baik pada dataset gambar, sekaligus mengurangi jumlah parameter yang terlibat [26].

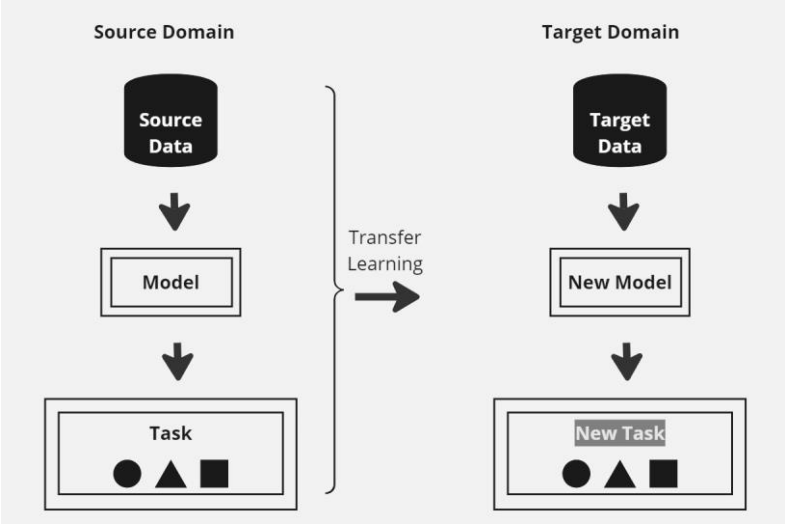
3. *Recurrent Neural Networks* (RNN)

Recurrent Neural Networks (RNN) adalah *neural network* yang dirancang untuk memproses data berkelanjutan atau data yang disajikan sebagai urutan untuk memanfaatkan kelanjutan data. Biasanya, data yang masuk ke setiap lapisan

tersembunyi berasal dari keluaran lapisan sebelumnya sebagai masukan ke lapisan saat ini, bersama dengan masukan tersembunyi. RNN dapat sangat menguntungkan dalam kasus-kasus dengan urutan masukan yang panjang, di mana persyaratan melibatkan mempertahankan konteks masukan yang sama tanpa mempengaruhi ukuran model yang digunakan. Oleh karena itu, RNN menjadi pilihan yang tepat untuk aplikasi NLP, meskipun informasi historis cenderung menghilang dalam jangka waktu yang lama, dan ini dapat memperlambat prosesnya [26].

2.2.3 *Transfer Learning*

Transfer learning adalah suatu teknik melakukan komputasi dengan menggunakan *pre-trained model* yang memungkinkan untuk mengadaptasi atau mentransfer pengetahuan yang diperoleh dari satu set tugas dan/atau domain ke set tugas dan/atau domain yang berbeda. Artinya, *pre-trained model* yang dilatih dengan sumber daya yang sangat besar, termasuk data, daya komputasi, waktu, dan biaya, yang dulunya merupakan sumber terbuka, dapat disetel dan digunakan kembali dalam pengaturan baru oleh komunitas teknik yang lebih luas dengan biaya yang lebih murah dari sumber daya aslinya persyaratan [27]. Berikut ilustrasi proses *Transfer learning* pada Gambar 2.2.

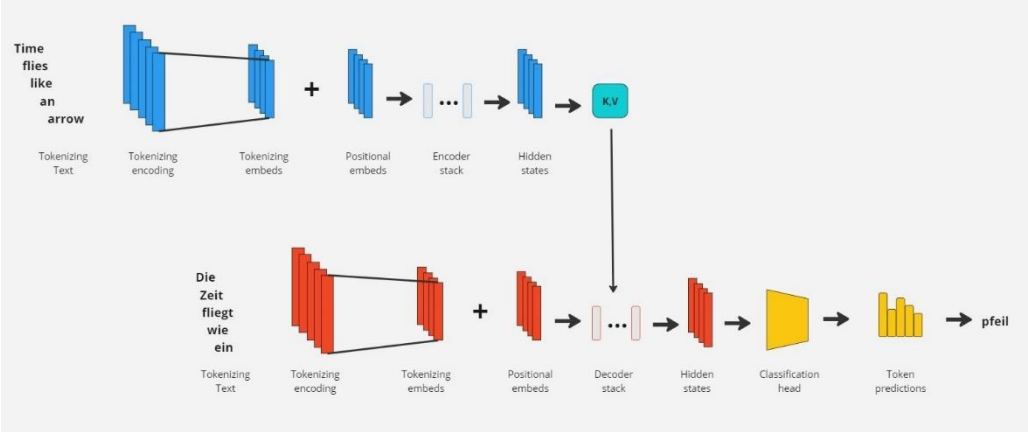


Gambar 2. 2 Ilustrasi proses *Transfer Learning* [27]

Ilustrasi proses *transfer learning* ditampilkan pada Gambar 2.2 dimulai dari proses di sebelah kiri menggambarkan proses *machine learning* tradisional, sementara proses di sebelah kanan mencerminkan proses *transfer learning*. Dapat dilihat bahwa *transfer learning* tidak hanya menggunakan data dari domain tugas target sebagai input untuk algoritma pembelajaran, tetapi juga memanfaatkan proses pembelajaran di domain sumber, termasuk data pelatihan, model, dan deskripsi tugas. Gambar 2.2 ini menggambarkan konsep utama dari *transfer learning*, yaitu mengatasi keterbatasan data pelatihan di domain target dengan memanfaatkan pengetahuan yang lebih banyak yang diperoleh dari domain sumber.

2.2.4 *Transformers*

Transformers didasarkan pada arsitektur *encoder-decoder* yang banyak digunakan untuk tugas-tugas seperti terjemahan mesin, di mana urutan kata-kata diterjemahkan dari satu bahasa ke bahasa lain. Arsitektur *transformer* meliputi *Encoder*, Mengubah urutan input token menjadi urutan vektor penyematan, yang sering disebut keadaan tersembunyi atau konteks. *Decoder*, menggunakan keadaan tersembunyi *encoder* untuk secara *iteratif* menghasilkan urutan keluaran token, satu token pada satu waktu [28].



Gambar 2. 3 *Arsitektur Encoder-Decoder Transformer* [28]

Arsitektur *transformer* pada awalnya dirancang untuk tugas-tugas urutan-ke-urutan seperti penerjemahan mesin, tetapi blok *encoder* dan *decoder* segera diadaptasi sebagai model mandiri. Meskipun ada ratusan model *transformator* yang berbeda, kebanyakan dari mereka termasuk dalam salah satu dari tiga jenis:

1. Encoder-only

Model-model ini mengubah urutan input teks menjadi representasi numerik yang kaya yang cocok untuk tugas-tugas seperti klasifikasi teks atau pengenalan entitas bernama. BERT dan variannya, seperti RoBERTa dan DistilBERT, termasuk dalam kelas arsitektur ini. Representasi yang dihitung untuk token yang diberikan dalam arsitektur ini bergantung pada konteks kiri yang ada di gambar (sebelum token) dan kanan (setelah token). Ini sering disebut dengan *bidirectional attention*.

2. Decoder-only

Penerapan *decoder* ada dalam Keluarga model GPT. Contoh diberikan sebuah perintah teks seperti "Terima kasih untuk makan siang, saya makan ..." model ini akan melengkapi urutan secara otomatis dengan secara berulang-ulang memprediksi kata yang paling mungkin.. Representasi yang dihitung untuk token yang diberikan dalam arsitektur ini hanya bergantung pada konteks kiri. Hal ini sering disebut dengan perhatian kausal atau *autoregresif.*

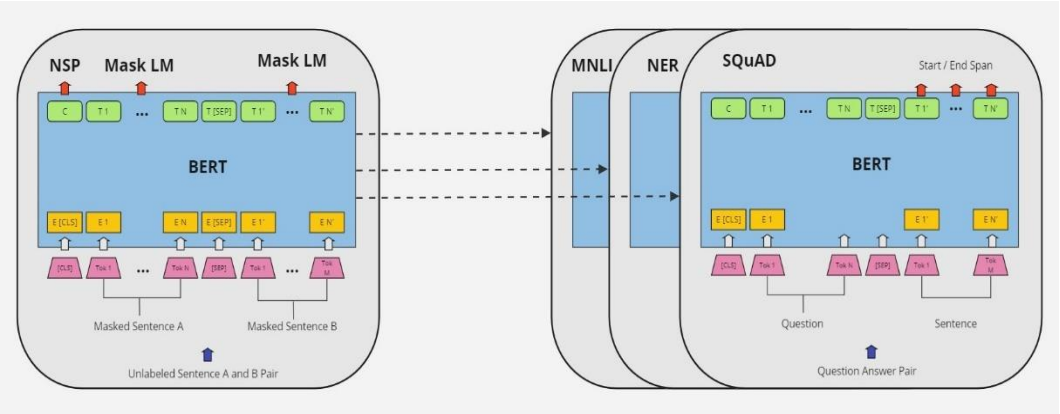
3. Encoder-Decoder

Encoder-Decoder digunakan untuk memodelkan pemetaan yang kompleks dari satu urutan teks ke urutan teks yang lain; mereka cocok untuk tugas penerjemahan dan peringkasan mesin. Dalam Selain arsitektur Transformer, yang seperti yang telah kita lihat menggabungkan *encoder* dan *decoder*, model BART dan T5 termasuk dalam kelas ini.

2.2.5 BERT

Bidirectional encoder representations from transformers (BERT) juga dinamai berdasarkan karakter *sesame street* yang populer sebagai anggukan terhadap tren yang dimulai oleh ELMo. Pada saat penulisan, variannya mencapai beberapa kinerja terbaik dalam mentransfer pengetahuan model bahasa yang telah dilatih sebelumnya ke tugas-tugas NLP. Model ini juga dilatih untuk memprediksi

kata-kata dalam urutan kata, meskipun prosedur masking yang tepat agak berbeda. Hal ini juga dapat dilakukan dengan cara yang tidak diawasi pada korpus yang sangat besar, dan bobot yang dihasilkan juga dapat digeneralisasi ke berbagai tugas NLP lainnya [27]. Berikut arsitektur BERT secara umum yang dapat dilihat pada Gambar 2.4.



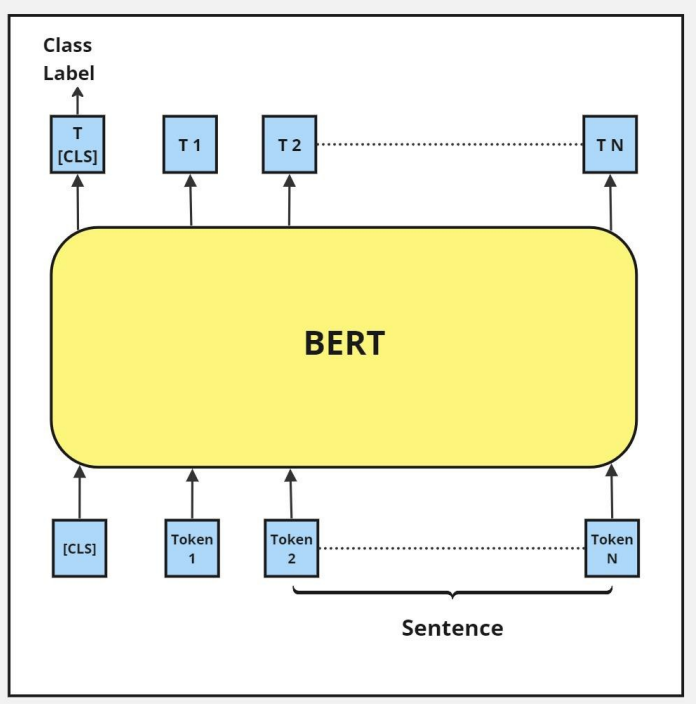
Gambar 2. 4 Arsitektur BERT [29]

BERT menggunakan dua strategi untuk mengatasi batasan arah tunggal. BERT mengalami pelatihan sebelumnya dalam dua tugas NLP, yaitu *masked language modeling* (MLM) dan *next sentence prediction* (NSP). MLM membantu melatih transformator secara dua arah dengan secara acak menyembunyikan beberapa token dalam teks input, sementara tugas NSP melatih representasi pasangan teks secara bersamaan. BERT meminimalkan fungsi kerugian gabungan dari kedua tugas selama pelatihan. Untuk menggunakan BERT, terdapat dua tahapan yang harus diikuti:

1. *Pre-Training*: Pada tahap ini, model dilatih pada data tanpa label melalui berbagai tugas prapelatihan.
2. *Fine-tuning*: Model BERT diinisialisasi dengan parameter dari tahap pra pelatihan dan kemudian disempurnakan dengan fine-tuning menggunakan data dari tugas-tugas spesifik, seperti klasifikasi, menjawab pertanyaan, dan lain sebagainya. Berikut tugas yang dapat diselesaikan dengan fine-tuning BERT antara lain:

a. *Single Classification*

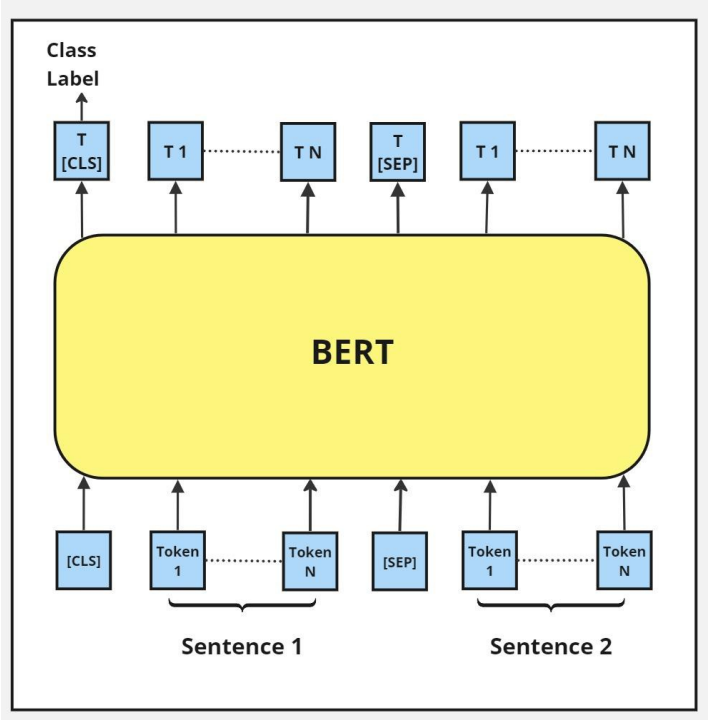
Single Classification Task, informasi tentang label klasifikasi terkandung dalam keluaran dari posisi 1. Vektor dari posisi pertama diteruskan ke *feedforward neural network*, dan kemudian fungsi *softmax* diterapkan untuk menghasilkan distribusi *probabilitas* untuk jumlah kelas yang terlibat dalam tugas, sebagaimana ditunjukkan pada Gambar 2.5 ini [10].



Gambar 2. 5 Arsitektur *Single Classification Task* [10]

b. *Sentence-Pair Text Classification*

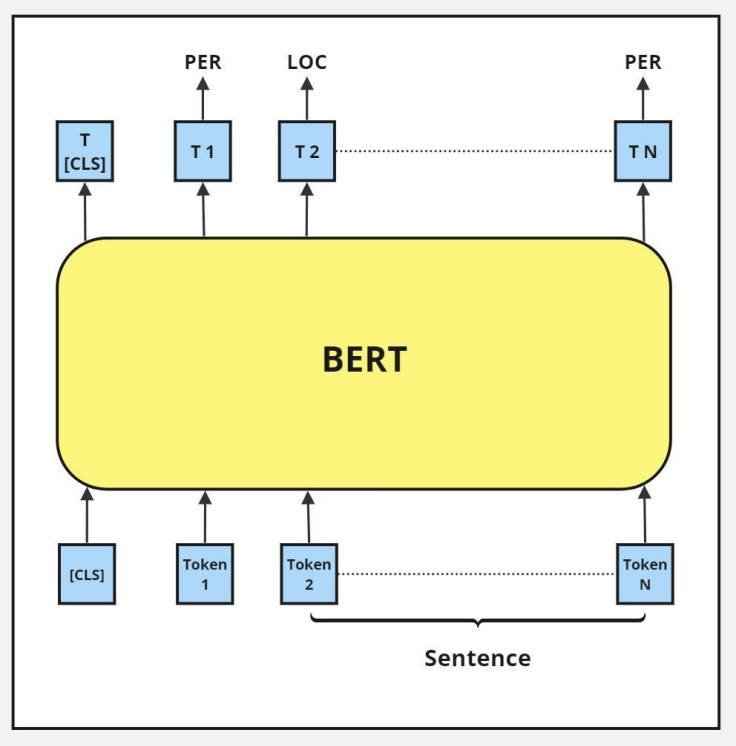
Sentence-Pair Text Classification Task perlu menentukan apakah kalimat kedua mengikuti kalimat pertama dalam sebuah korpus atau kalimat kedua merupakan jawaban dari kalimat pertama (pertanyaan). Tugas ini merupakan klasifikasi yang memerlukan jawaban Ya atau Tidak. Hasil dari tugas-tugas ini dapat ditentukan menggunakan vektor keluaran pertama, sebagaimana diilustrasikan pada gambar 2.6 ini [10].



Gambar 2. 6 Arsitektur *Sentence-Pair Text Classification Task* [10]

c. *Named Entity Recognition* (NER)

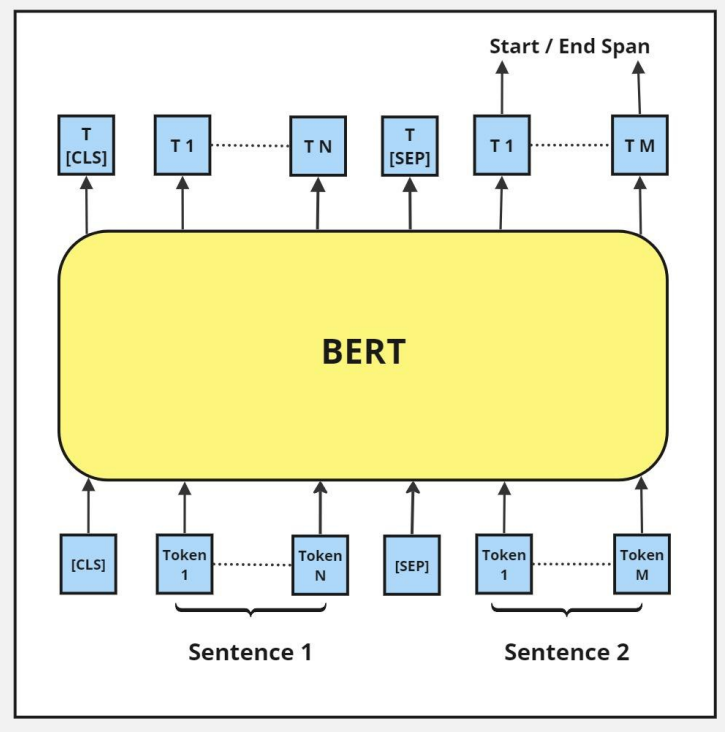
Named Entity Recognition (NER) *Task,* memerlukan model untuk *mengetahui* apakah token dalam sebuah kalimat merujuk pada seseorang, lokasi, tanggal, dan sebagainya. Setiap token harus mengatakan entitas mana yang dilayani dari entitas yang telah kita sebutkan. Ini adalah tugas Seq2Seq di mana ukuran input harus sama dengan ukuran output. Model BERT mengeluarkan sebuah vektor untuk setiap posisi. Sekarang, setiap output posisi ini dapat diumpankan ke *neural network* untuk mengetahui entitas bernama untuk token tertentu. Hal ini diilustrasikan dalam Gambar 2.7 [10].



Gambar 2. 7 Arsitektur *Named Entity Recognition* (NER) *Task* [10]

d. *Question Answering*

Question Answering Task, BERT melakukan ekstraksi token dari *pertanyaan* dan bagian teks, lalu menggabungkannya sebagai masukan. BERT memulai dengan token [CLS] yang menandai awal kalimat, dan menggunakan token pemisah [SEP] untuk memisahkan pertanyaan dan teks bacaan. *Hidden layer* terakhir dari BERT kemudian diubah dan digunakan *softmax* untuk menghasilkan distribusi *probabilitas* untuk indeks awal dan akhir dari substring dalam kalimat teks masukan yang menyatakan jawaban. Hal ini diilustrasikan dalam Gambar 2.8.



Gambar 2. 8 Arsitektur *Question Answering Task* [10]

2.2.6 *Simple Transformers*

Simple Transformers adalah *library* yang dibangun di atas arsitektur transformers untuk menyelesaikan tugas-tugas hilir seperti klasifikasi teks biner atau multi-kelas. *Library* ini telah dikembangkan untuk menyertakan model penjawab pertanyaan, pengenalan entitas bernama, dan generasi bahasa. Model penjawab pertanyaan dapat dilatih, dievaluasi, dan diuji menggunakan berbagai parameter yang sesuai dengan tugas-tugas tertentu dan dapat meningkatkan kinerja.

Selama pelatihan, parameter dapat difokuskan seperti *batch size*, *learning rate*, dan jumlah *epoch* pada tugas question answering. Sedangkan untuk prediksi, user dapat mengatur model untuk mengembalikan lima jawaban untuk setiap pertanyaan untuk contohnya. Output dari model adalah dua daftar yang berisi jawaban dan nilai probabilitasnya [30]. *Simple transformers* sendiri merupakan *library* NLP yang dirancang untuk menyederhanakan penggunaan model transformers tanpa harus mengorbankan utilitas. Dan dibangun di atas karya luar

biasa dari Hugging Face dan *transformers library*. *Simple transformers* dirancang sesuai dengan cara seseorang biasanya menggunakan model *transformers*.

Simple transformers bercabang menjadi tugas-tugas NLP yang umum seperti *classification text, question answering*, dan *language model* pada tingkat tertinggi. Masing-masing tugas NLP ini memiliki model *transformers* sederhana yang khusus pada tugasnya. Meskipun semua model khusus tugas mempertahankan pola penggunaan yang konsisten (inisialisasi, pelatihan, evaluasi, prediksi), pemisahan ini memungkinkan kebebasan untuk menyesuaikan model dengan kasus penggunaan spesifik tertentu.

2.2.7 *Haystack*

Haystack adalah sebuah *framework* yang dikembangkan dalam *python* yang memungkinkan *pipeline* yang kuat dan siap produksi untuk berbagai kasus penggunaan pencarian. *Haystack* berisi berbagai opsi pencarian dan dapat memberikan pengalaman pencarian yang unik dengan memungkinkan pengguna untuk menerapkan model NLP yang canggih.

Haystack sendiri merupakan framework *python* bersifat *open-source* yang dikembangkan oleh *deepset* untuk membangun aplikasi khusus dengan *large language models* (LLM). Hal ini memungkinkan user dengan cepat mencoba model-model terbaru dalam NLP sekaligus menjadikannya fleksibel dan mudah digunakan. *Haystack* sekarang menjadi framework yang lengkap untuk membangun aplikasi NLP yang siap produksi berkat komunitas pengguna dan developer yang inspiratif [20].

2.2.8 Exact Match (EM)

Exact Match (EM) adalah metrik biner yang memeriksa apakah prediksi cocok dengan jawaban yang sebenarnya. Meskipun metrik ini bekerja dengan baik untuk jawaban faktual yang pendek, seperti nama orang atau lokasi, metrik ini memiliki beberapa kelemahan yang jelas ketika harus membandingkan jawaban pendek yang sedikit berbeda atau jawaban yang lebih panjang dan rumit. Bahkan prediksi yang berbeda dari kebenaran dasar hanya dalam satu karakter dalam string jawaban dievaluasi sebagai sepenuhnya salah [31].

2.2.9 *Confusion Matrix*

Confusion Matrix atau *Matrix Error* merupakan *matrix* evaluasi yang yang menggambarkan kinerja model klasifikasi pada satu set data uji [32]. *Confusion matrix* adalah matriks dua dimensi, diindeks dalam satu dimensi oleh kelas sebenarnya dari sebuah objek dan di dimensi lainnya oleh kelas yang ditetapkan oleh pengklasifikasi. Kasus khusus dari *Confusion matrix* sering digunakan dengan dua kelas, satu kelas ditunjuk sebagai kelas positif dan kelas negatif. Dalam konteks ini, empat sel matriks ditetapkan sebagai *true positive (TP), false positive (FP), true negative (TN), dan false negative (FN)* [33].

Tabel 2. 2 Tabel Confusion Matrix

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Hasil dari *confusion matrix* dapat digunakan untuk mengukur evaluasi seperti *accuracy*, *precision*, *recall* dan *f1-score*.

Accuracy merepresentasi jumlah prediksi yang benar terhadap total keseluruhan data dan dihitung dengan formula (2.1).

$$Accuracy = \frac{(TP + TN)}{(TN + FP + TP + FN)} \tag{2.1}$$

Precision merepresentasi akurasi prediksi positif yang dibuat oleh model, dan didapatkan dengan rumus (2.2).

$$Precision = \frac{TP}{(TP + FP)} \tag{2.2}$$

Recall mengukur kemampuan model untuk menangkap semua contoh yang relevan dalam set data dengan memakai rumus (2.3).

$$Recall = \frac{TP}{\left(TP + FN\right)} \tag{2.3}$$

F1-Score adalah rata-rata harmonik dari presisi dan recall, yang memberikan ukuran yang seimbang dari kinerja model [33]. Dan untuk mendapatkannya dihitung menggunakan rumus (2.4).

$$F1 - score = 2 * \frac{\left(precision * recall\right)}{\left(precision + recall\right)} \tag{2.4}$$