# BAB II

# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

| No | Peneliti | Judul | Metode | Hasil |
|---|---|---|---|---|
| 1. | (Rauf et al., 2019) | Using BERT for Checking the Polarity of Movie Reviews. | Bidirectional *Encoder* Representations from *Transformers*s | Hasil evaluation *accuracy* yang diperoleh mencapai 0.89 dengan loss 0.4856, *precision* 0.9174 dan *recall* 0.8812 |
| 2. | (Albert, 2023) | Analisis Topik dan Perbandingan Klasifikasi pada Kolom Komentar Video Youtube Edukasi Indonesia Menggunakan Pendekatan Latent Dirichlet Allocation | Pemodelan Topik Latent Dirichlet Allocation (LDA) | Hasil dari penelitian dapat disimpulkan bahwa LDA dapat menentukan kata kunci dan topik dengan akurat. |

| | | | | |
|---|---|---|---|---|
| 3. | (Putri, 2020) | Analisis Sentimen Review Film Berbahasa Inggris Dengan Pendekatan Bidirectional *Encoder* Representations from *Transformers*s | Bidirectional *Encoder* Representations from *Transformers*s | Hasil akurasi menggunakan BERT adalah 73% dengan 2000 review dimana 1000 reiew dengan sentimen positif, dan 1000 review dengan sentimen negatif |
| 4. | (Azzahra et al., 2024) | Sentiment Analysis during Jakarta Flood for Emergency Responses and Situatoional Awareness in Disaster Management using BERT | Bidirectional *Encoder* Representations from *Transformers*s | Hasil akurasi yang diperoleh saat training *dataset* adalah 90% dan test *dataset* adalah 79%. |
| 5. | (Kurniawan R & Zufria, 2022) | Penerapan Text Mining Pada Sistem Penyeleksian Judul Skripsi Menggunakan Algoritma Latent Dirichlet Allocation(LDA) | Algoritma Latent Dirichlet Allocation(LDA) | Hasil dari penelitian ini bahwa algoritma Latent Dirichlet Allocation dapat digunakan untuk menentukan topik dari judul skripsi yang diajukan mahasiswa. Mahasiswa dapat mengetahui secara |

| | | | | |
|---|---|---|---|---|
| | | | | langsung kesesuaian topik dan peluang diterimanya setiap judul proposal yang diajukan. Dengan penggunaan sistem ini, proses Indonesian |
| 6. | (Nurmawati & Amanda, 2023) | Analisis Sentimen Dan Pemodelan Topik Pada *Tweet* Terkait Data Badan Pusat Statistik | Pemodelan Topik Latent Dirichlet Allocation (LDA) | Hasil pemodelan topik pada setiap tahun menghasilkan jumlah topik yang berbeda sesuai dengan nilai coherence tertinggi. Pada tahun 2020, jumlah topik yang dihasilkan adalah sebanyak 5 topik, dengan kemunculan topik tertinggi yaitu indikator kemiskinan dan indikator sensus penduduk yang berada pada ragam data Statistik Sosial. |
| 7. | (Putu et al., 2021) | Analisis Sentimen dan Pemodelan Topik Pariwisata Lombok Menggunakan Algoritma Naive | Algoritma Naive Bayes dan Latent Dirichlet Allocation | Hasil analisis sentimen diperoleh 9.496 *tweet* dengan pembagian 8.996 *tweet* sentimen positif dan 500 *tweet* sentimen negatif, |

| | | | | |
|---|---|---|---|---|
| | | Bayes dan Latent Dirichlet Allocation | | sehingga dapat disimpulkan bahwa lebih banyak wisatawan memberikan respon positif daripada negatif terhadap pariwisata Lombok. |

## 2.2 Kajian Literatur

Konflik antara Ukraina dan Rusia ini mengakibatkan korban jiwa dan kerusakan besar pada infrastruktur fisik di Ukraina. Imbasnya timbul gelombang pengungsi dengan lebih dari 1 juta pengungsi yang terpaksa bermigrasi ke negara-negara tetangga. Kondisi ekonomi pun berfluktuasi sehingga harga energi dan komoditas termasuk gandum dan biji-bijian lainnya telah melonjak, menambah tekanan inflasi dari gangguan rantai pasokan dan rebound dari masih berlangsungnya pandemi Covid-19. Jika eskalasi konflik lagi maka kerusakan ekonomi akan lebih dahsyat. Sanksi terhadap Rusia juga akan berdampak besar pada ekonomi global dan pasar keuangan, dengan limpahan yang signifikan ke negara lain. Di banyak negara, krisis menciptakan kejutan yang merugikan baik terhadap inflasi maupun aktivitas, di tengah tekanan harga yang sudah meningkat. Otoritas moneter perlu secara hati-hati memantau kenaikan harga internasional terhadap inflasi domestik, untuk mengkalibrasi respons yang tepat. Krisis ini akan menciptakan kompromi kebijakan yang kompleks, yang semakin memperumit lanskap kebijakan seiring pemulihan ekonomi dunia dari krisis pandemi (kompastv.com).

## 2.3 Natural Language Processing (NLP)

Natural Language Processing atau Pengolahan Bahasa Alami adalah salah satu cabang ilmu Kecerdasan Buatan yang mempelajari dan mengembangkan bagaimana komputer dapat mengerti, memahami, dan memproses bahasa alami dalam bentuk teks atau tuturan kata. NLP menganalisa bahasa manusia sedemikian rupa sehingga komputer dapat memahami bahasa alami seperti halnya manusia (Soyusiawaty, n.d.). NLP adalah salah satu bidang antar disiplin yang menggabungkan komputasi linguistik, ilmu komputasi, ilmu kognitif, dan kecerdasan buatan. Pada umumnya, NLP banyak diaplikasikan di berbagai hal seperti speech recognition, pemahaman bahasa lisan, sistem dialog, analisis leksikal, mesin penerjemah, knowledge graph, analisis sentimen, sistem pintar dan peringkasan bahasa alami.

Sebuah sistem NLP dapat dimulai dari tingkat kata untuk menentukan struktur dan sifat morfologis (seperti part-of-speech atau makna) dari kata; kemudian dapat beralih ke tingkat kalimat untuk menentukan urutan kata, tata bahasa, dan arti dari seluruh kalimat. Kemudian ke konteks dan keseluruhan domain. Kata atau kalimat yang diberikan mungkin memiliki makna atau konotasi yang berbeda dalam konteks tertentu, yang terkait dengan banyak kata atau kalimat lain dalam konteks yang diberikan.

## 2.4 Analisis Sentimen

Analisis sentimen merupakan salah satu bidang penelitian komputasi yang mempelajari opini-opini, sentimen-sentimen, dan emosi-emosi yang ada pada teks. Analisis sentimen yang juga dikenal dengan nama opinion mining, telah menjadi salah satu topik hangat di bidang NLP dan data mining (penambangan data). Tujuan utama dari analisis sentimen adalah untuk memproses, mengekstrak, merangkum, dan menganalisa informasi yang ada dalam teks melalui metode yang berbeda beda, sehingga dapat menyimpulkan emosi dan sudut pandang yang diberikan oleh peneliti dari teks tersebut, dan membagi kecenderungan emosional di teks melalui informasi subjektif yang terkandung di dalamnya. Sentimen sendiri didefinisikan sebagai suatu sikap positif atau negatif seseorang atau sekelompok orang yang diarahkan kepada sesuatu. Opini atau sentimen dapat direpresentasikan sebagai quintuple yang terdiri dari ei, aij, sijkl, hk, tℓ dimana ei

adalah entiti atau target dari opini, aij adalah aspek dari target opini, sijkl adalah opini atau sentimen yang diberikan ke target, hk adalah opinion holder atau pemberi opini, dan tℓ adalah waktu ketika opini diberikan (Clarita, 2023).

Teks-teks opini termasuk ke dalam data yang tidak terstruktur (unstructured data), sehingga perlu dilakukan preprocessing untuk membuat data tersebut menjadi terstruktur dan dapat diproses untuk mengambil aspek yang ada melalui *Tokenisasi*, word segmentation, Part-of-Speech Tagging, stemming, dan lain-lain(Zulfa & Winarko, 2017) .

Secara umum, analisis sentimen dibagi menjadi tiga tingkatan yaitu tingkat dokumen (*document level*), tingkat kalimat (*sentence level*), dan tingkat berbutir halus (*fine-grained level*). *Document level* dan *sentence-level* dapat pula dikategorikan ke dalam coarse-grained level. Metode dalam analisis sentimen terbagi menjadi dua jenis, yaitu learning-based dan lexical-based. Learning-based menggunakan data training dan data testing, sedangkan lexical-based menggunakan kamus (opinion lexicon) (Alwasi'a A, 2020).

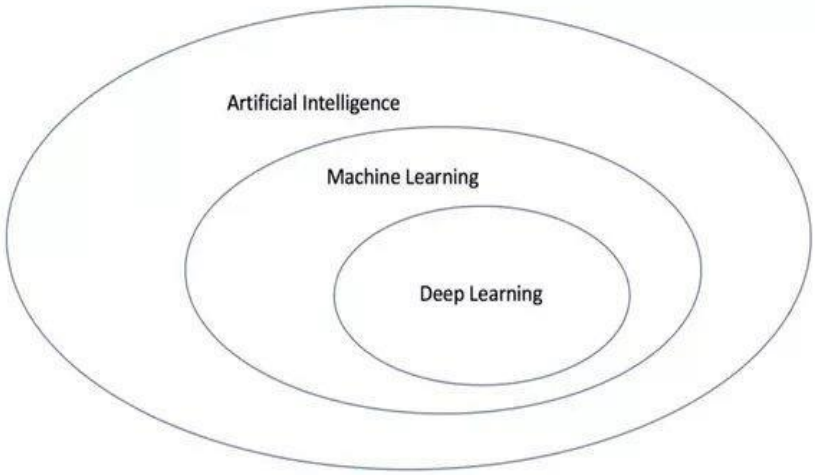
## 2.5 Machine Learning

Machine learning atau pembelajaran mesin adalah salah satu bidang ilmu di Kecerdasan Buatan. Machine learning, sesuai dengan namanya, bertujuan untuk membuat mesin dilatih dengan banyak contoh atau *dataset* yang berhubungan dengan tugas yang dibutuhkan. Mesin mempelajari pola-pola yang diberikan berdasarkan *dataset* dan menghasilkan sebuah rule sendiri. Sehingga ketika suatu data dimasukkan ke dalam mesin, mesin sudah dapat mengenali data tersebut. Secara umum, machine learning terbagi menjadi empat kategori besar yaitu supervised learning, unsupervised learning, self-supervised learning, dan reinforcement learning . Supervised learning adalah pendekatan yang paling sering digunakan. Supervised learning membuat mesin belajar dari *dataset* yang sudah diberi label atau anotasi. Sedangkan unsupervised learning merupakan kebalikannya, dengan memberikan *dataset* yang tidak diberi label. Selfsupervised learning adalah sebuah supervised learning tetapi tanpa *dataset* yang dilabeli oleh annotator. *Dataset* yang digunakan tetap menggunakan label akan tetapi label diperoleh dari input data yang menggunakan algoritma heuristic (Chollet, 2018).

Algoritma yang sering digunakan pada machine learning antara lain K-Nearest Neighbor, Naïve-Bayes, Support Vektor Machine, K-Means, dan lain-lain.

## 2.6 Deep Learning

*Deep Learning* adalah cabang dari *machine learning* yang merupakan bagian dari Kecerdasan Buatan. *Deep learning* merupakan *neural network* yang lebih modern dan bersifat *deep* atau mendalam karena memiliki jauh lebih banyak *layers* dibandingkan dengan *neural network* pada biasanya (Chollet, 2018). Kata *"deep"* mengacu pada jumlah *hidden layers* yang ada, semakin banyak *layers*nya, maka semakin *"deep"* pembelajaran yang dilakukan oleh jaringan. *Deep learning* bekerja untuk mempelajari sehingga tidak hanya dapat memprediksi tetapi juga merepresentasikan data dengan benar, sehingga cocok untuk melakukan prediksi (Mahler et al., n.d.). Deep learning dapat dibagi ke dalam tiga metode pendekatan yaitu *supervised*, *semi-supervised*, dan *unsupervised learning*. *Deep learning* didukung oleh banyak *framework* seperti Torch, Theano, TensorFlow, dan lain-lain.




**Gambar 2. 1 Hubungan antara Kecerdasan Buatan, Machine Learning, dan Deep Learning**

## 2.7 *Bidirectional Encoder Representations from Transformers* (BERT)

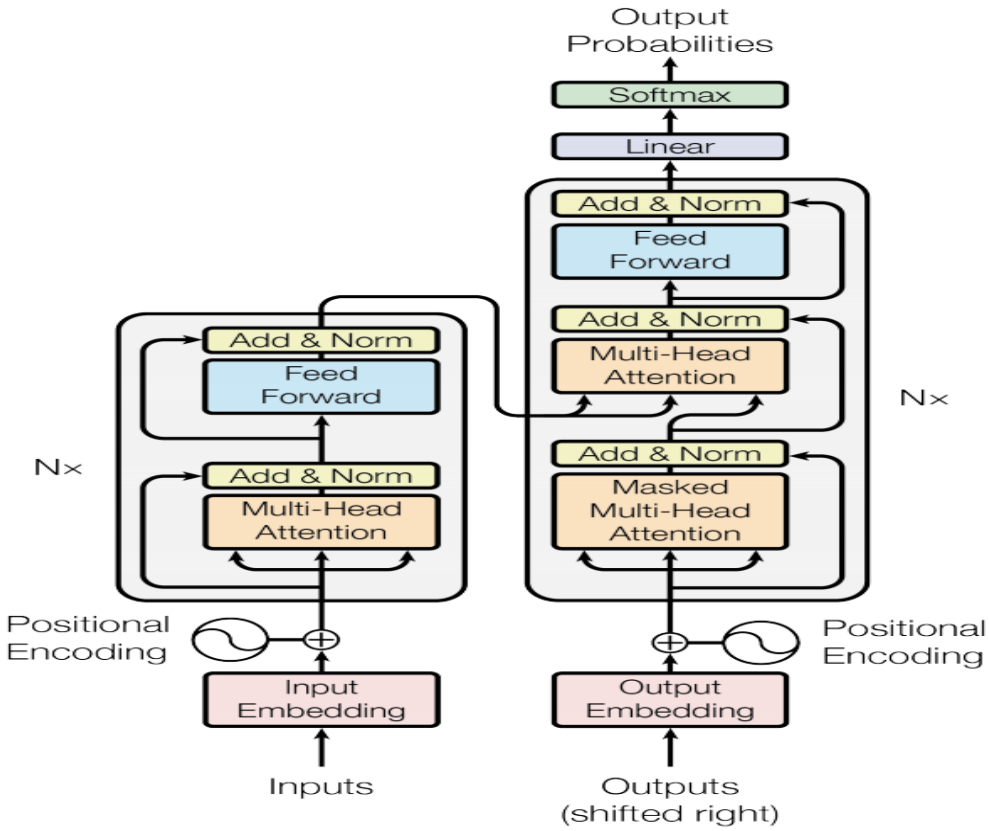
*Bidirectional Encoder Representations from Transformers* (Devlin et al., 2019) atau disingkat BERT adalah model representasi bahasa terlatih yang dikembangkan oleh para peneliti di Google AI Language pada tahun 2018. BERT dikembangkan berdasarkan teknik-teknik *deep learning* dan berbagai metode seperti *semisupervised learning*, ELMo, ULMFiT, OpenAI *Transformers*s, dan *Transformers*s. Sesuai dengan namanya, BERT menggunakan *Transformers*. *Transformers* adalah sebuah mekanisme yang mempelajari hubungan kontekstual antara kata-kata dalam teks (Munikar et al., n.d.). *Transformers* dapat memahami dan mengkonversi pemahaman yang diperoleh dengan mekanisme yang bernama *self-attention mechanism*. *Self-attention mechanism* adalah cara *Transformers* untuk mengubah "pemahaman" kata terkait lainnya menjadi kata-kata yang akan diproses dengan mekanismenya. Pada *Transformers* terdapat dua mekanisme, yaitu:

a. *Encoder*

*Encoder* berfungsi untuk membaca seluruh input teks sekaligus. *Encoder* terdiri dari stack (tumpukan) dari N = 6 *layerss* yang identik. Setiap *layers* memiliki dua sub-*layers* yaitu self-attention *layers* dan feed-forward neural network. Dengan self-attention *layers*, *Encoder* dapat membantu node untuk tidak hanya fokus kepada kata yang sedang dilihat tetapi juga untuk mendapatkan konteks semantik dari kata tersebut. Setiap posisi di *Encoder* dapat menangani semua posisi di *layers* sebelumnya di *Encoder*.

b. *Decoder*

*Decoder* berfungsi untuk menghasilkan urutan *output* yang berupa prediksi. *Decoder* juga terdiri dari stack (tumpukan) dari N = 6 *layerss* yang identif. Setiap *layers* terdiri dari dua sub-*layers* seperti yang ada pada *Encoder*, dengan tambahan *attention layers* di antara dua *layerss* tersebut untuk membantu node saat ini mendapatkan *key content* yang membutuhkan *attention* (Pratama & Romadhony, 2020) dengan melakukan *multi-head attention* pada *output* dari *Encoder*. Sama dengan di *Encoder*, *self-attention layers* di *decoder* membuat setiap posisi di *decoder* dapat menangani semua posisi sebelumnya dan posisi saat itu.

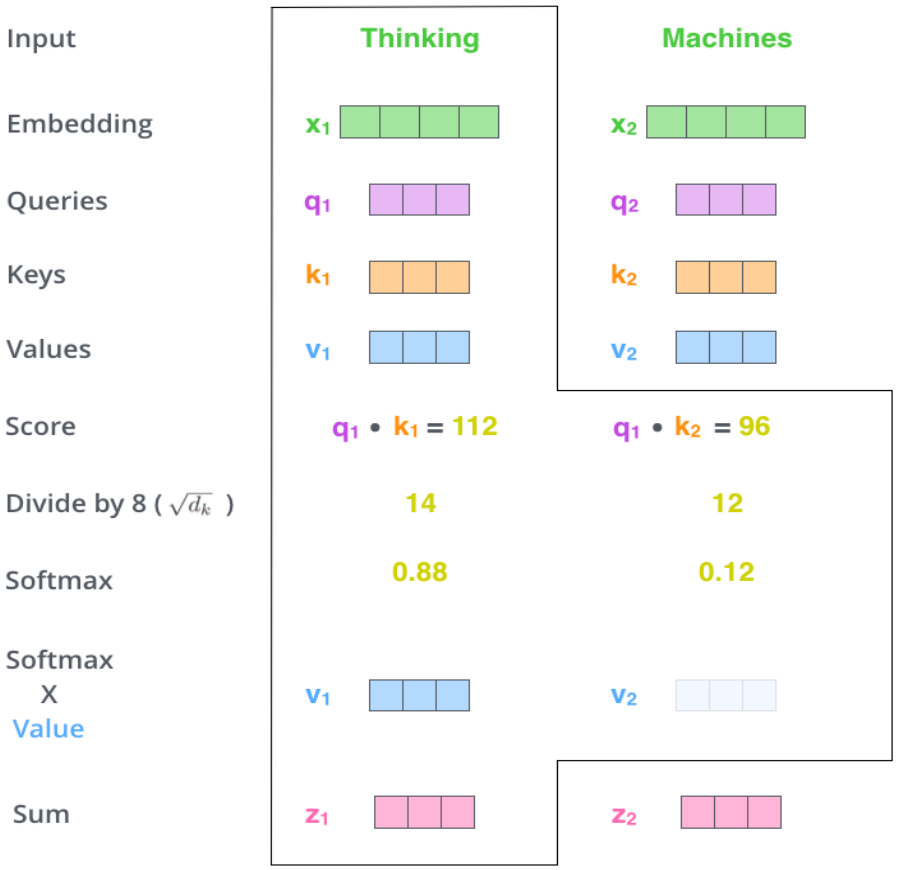



**Gambar 2. 2** *Encoder* (kiri) dan Decoder (kanan) (Vaswani et al., 2017)

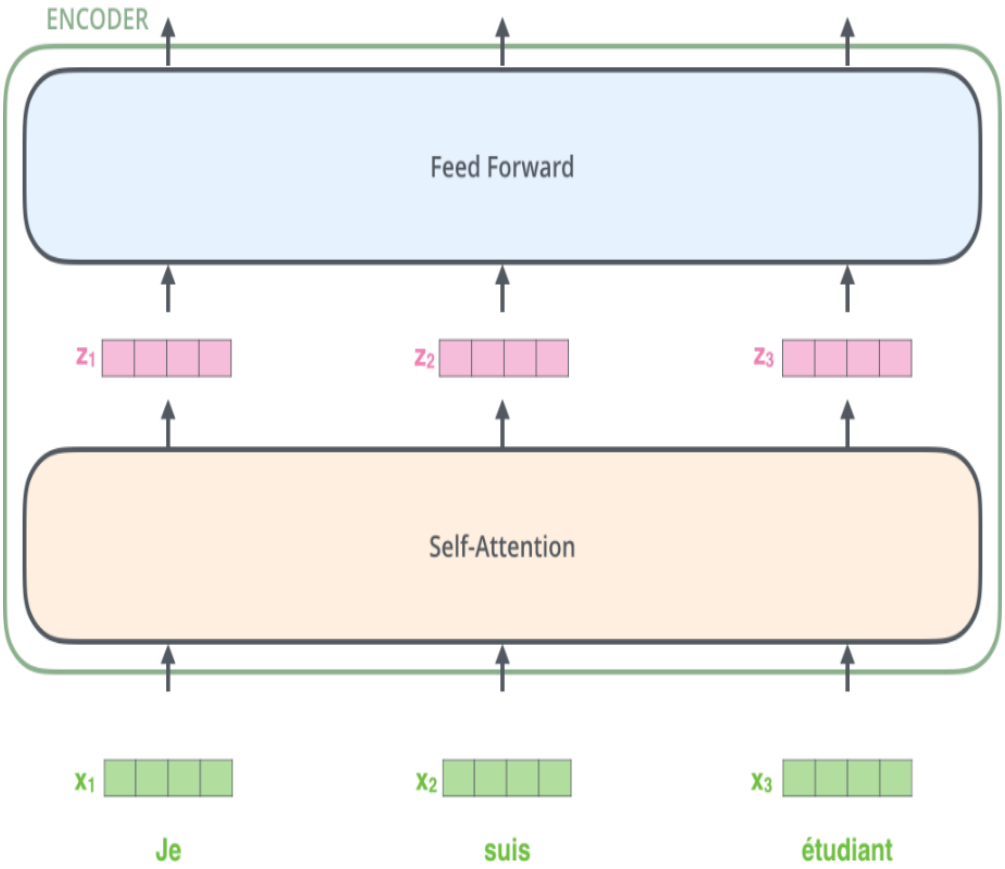
Langkah-langkah berikut menunjukkan proses yang terjadi pada *Encoder* dan decoder (Alammar, 2018):

1. Setiap input kata yang memasuki *Encoder* diubah menjadi sebuah *list vektor* menggunakan *embeddings*. Karena *self-attention layers* tidak membedakan urutan kata-kata pada sebuah kalimat, *positional encoding* ditambahkan untuk menunjukkan posisi dari tiap kata. Tiap vektor dari input kata memiliki ukuran 512. Proses ini hanya terjadi di *Encoder* yang berada paling bawah, sehingga *Encoder* lainnya akan menerima *output* dari *Encoder* yang pertama.
2. Input vektor melewati dua *layers* yang ada pada tiap *Encoder* yaitu *self-attention layers* dan *feed-forward neural network*. Pada *self-attention layers* dibuat tiga vektor dari masing-masing input vektor yaitu *Query*, *Key*, dan Value vektor. Ketiga vektor ini dibuat dengan mengalikan *embedding*. Dimensi dari tiap vektor adalah 64. Setelah itu, nilai *self-attention* dari tiap kata dihitung dengan mengalikan *query vektor* dan *key vektor* seperti yang ada pada Gambar 2.4.

Kemudian, nilai *self-attention* dibagi 8 karena 8 adalah akar kuadrat dari dimensi tiap vektor yaitu 64. Nilai *self-attention* juga dihitung dengan *softmax* sehingga tiap *value vektor* akan dikali dengan nilai dari *softmax*. Akhirnya *value vektor* dijumlahkan dan menjadi *output* dari *self-attention layers*. Output dari *self-attention layers* kemudian masuk ke *feed-forward* untuk tiap posisi seperti yang tertera pada Gambar 2.5.



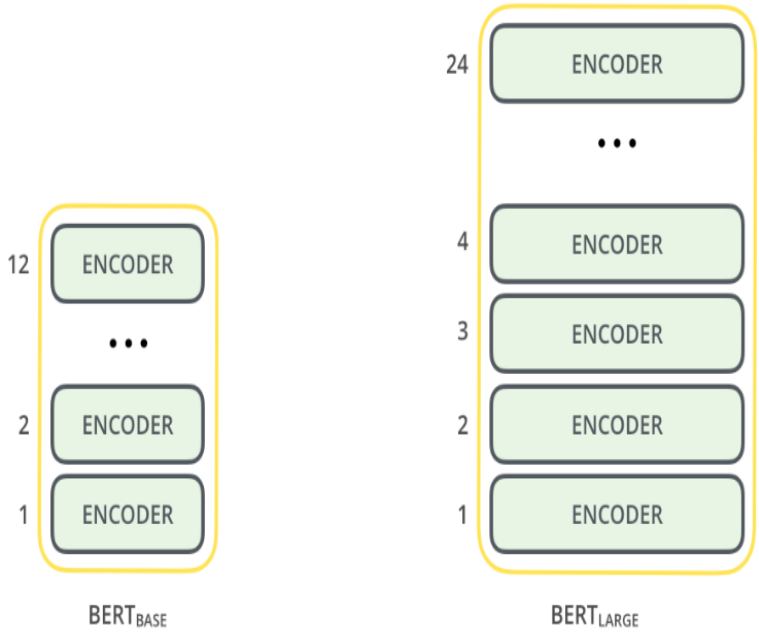

**Gambar 2. 3** Proses pada Self-attention *Layers*




**Gambar 2. 4** Proses pada *Encoder* (Alammar, 2018)

3. Setelah setiap proses pada *Encoder* selesai, *output* dari *Encoder* yaitu *vektor key* dan *vektor value* kemudian memasuki *decoder*. Tiap input dan *output* dari *selfattention layers* dan *feed-forward neural network* di *Encoder* dan *decoder* diproses oleh *layers add & norm* yang berisi struktur residual dan normalisasi *layers*. Proses yang terjadi pada *decoder* sama dengan *Encoder* akan tetapi di antara *self-attention layers* dan *feed-forward neural network* terdapat *attention layers* yang membantu *decoder* untuk fokus pada bagian-bagian dari kata yang relevan. *Self-attention layers* di *decoder* hanya boleh untuk menghadiri posisi sebelumnya dari *output*. *Output* dari tiap langkah dimasukkan ke dalam *decoder* terus menerus dan hasil dari *decoder* sama seperti hasil dari *Encoder*. Akhirnya, *output* dari tumpukan *decoder* menghasilkan sebuah *vektor* dengan *nilai float*. Untuk mengubahnya menjadi sebuah kata-kata, *layers* tambahan berupa *fully connected layers* dibutuhkan beserta *softmax layers*.

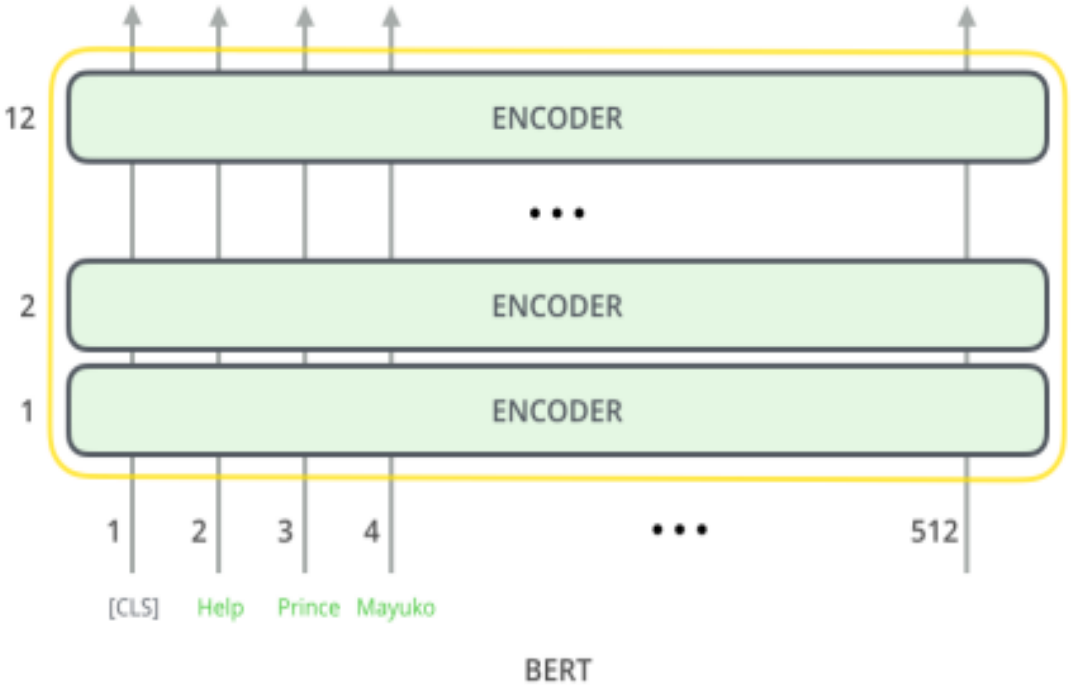
Arsitektur model BERT berupa *multi-layers bidirectional Transformers* seperti yang dilakukan pada implementasi asli *Transformers* tetapi hanya menggunakan proses sampai *Encoder* saja. Pada implementasinya, terdapat dua ukuran model yang ada pada BERT, yaitu $BERT_{BASE}$ dan $BERT_{LARGE}$. Kedua ukuran model BERT ini memiliki banyak lapisan *Encoder* atau *Transformers* Blocks. $BERT_{BASE}$ memiliki *Encoder* dengan 12 *layerss*, 12 *self-attentions heads*, *hidden size* sebesar 768, dan 110M *parameters*. Sedangkan $BERT_{LARGE}$ terdapat 24 *layerss*, 16 *selfattention heads*, *hidden size* sebesar 1024, dan 340M *parameters*. $BERT_{BASE}$ dilatih selama 4 hari menggunakan 4 cloud TPUs sedangkan $BERT_{LARGE}$ membutuhkan 4 hari menggunakan 16 TPUs.



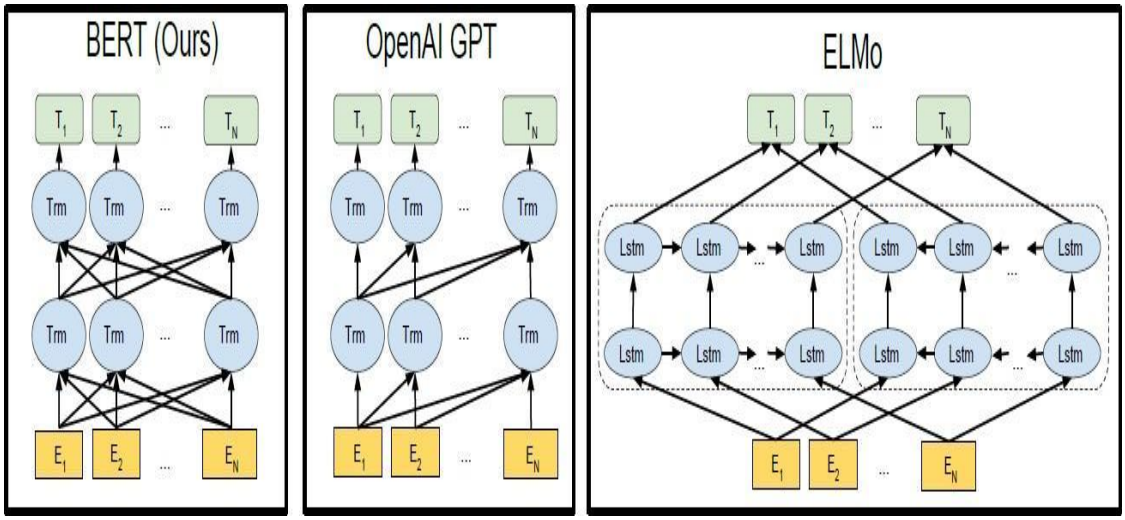

**Gambar 2. 5** Perbedaan Ukuran $BERT_{BASE}$ dan $BERT_{LARGE}$

Sesuai dengan namanya, BERT hanya menggunakan *Encoder*. Sehingga arsitektur BERT terlihat seperti Gambar 2.7. BERT berbeda dengan model terarah (*directional*) yang melihat urutan teks dari kiri-ke-kanan, kanan-ke-kiri, atau gabungan dari kiri-ke-kanan dan kanan-ke-kiri. Model bahasa yang dilatih secara *bidirectional* dapat memiliki pemahaman yang lebih dalam tentang konteks daripada model bahasa satu arah. Gambar 2.8 menunjukkan perbandingan antara arsitektur BERT dengan OpenAI GPT dan ELMo. Di antara ketiga model

arsitektur tersebut, hanya BERT yang secara bersamaan melihat kepada konteks kiri dan kanan di setiap *layers*nya.




**Gambar 2. 6** Arsitektur BERT




**Gambar 2. 7** Perbedaan Antara Arsitektur BERT Dengan OpenAI GPT dan ELMo.

BERT menggunakan WordPiece embeddings dengan 30,000 token *vocabulary*. Token pertama dari tiap urutan selalu berupa token klasifikasi khusus yaitu [CLS]. BERT dapat dilatih untuk memahami sebuah bahasa dan dapat pula disempurnakan (*fine-tune*) untuk mempelajari tugas-tugas tertentu. *Training* di BERT terdiri dari dua tahap, *pre-training* dan *fine-tuning*. Tahap pertama yaitu *pre-training* adalah tahap dimana BERT dibuat untuk memahami dan mempelajari bahasa dan konteksnya. BERT dapat memahami dengan *training* dengan dua tugas *unsupervised* yang dilakukan bersamaan yaitu Masked Language Model dan Next Sentence Prediction.

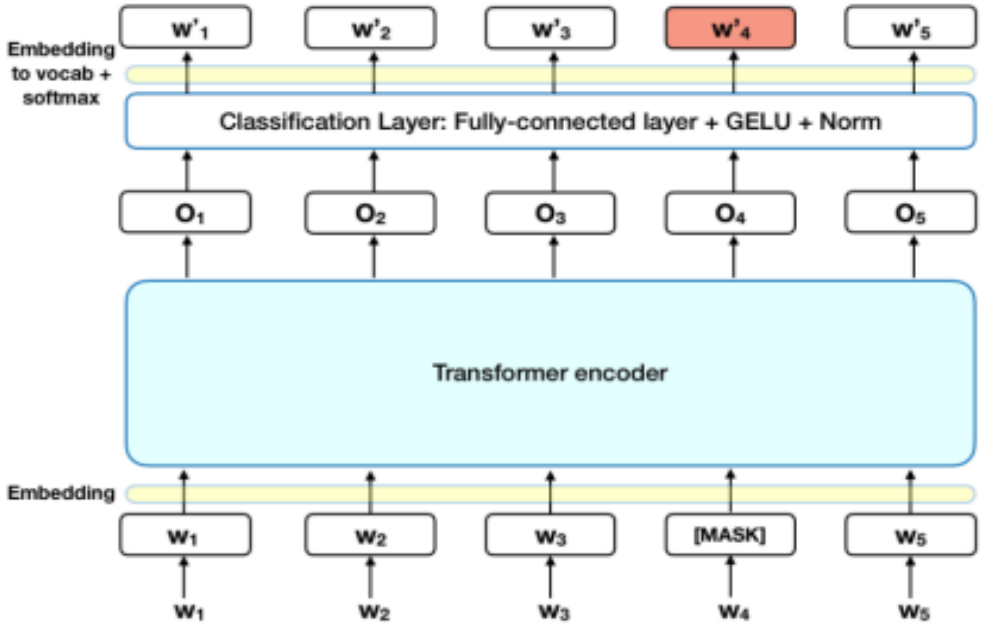
1. *Masked Language Modelling* (*Masked LM*)

   Tujuan dari *Masked Language Modelling* adalah untuk memberi *mask* atau penutup ke kata secara acak pada kalimat dengan probabilitas yang kecil. Sebelum memasukkan urutan kata ke dalam BERT, 15% dari kata-kata di tiap urutan kata diganti dengan token [MASK]. Kemudian model akan mencoba untuk memprediksi nilai asli dari kata yang diberi [MASK] berdasarkan konteks yang diberikan oleh kata lain yang tidak ditutup dengan [MASK] di dalam urutan kata. Secara teknis, prediksi kata-kata *output*:

   i) Membutuhkan lapisan klasifikasi di atas *output Encoder*

   ii) Mengalikan *vektor output* dengan matriks *embedding* kemudian mengubahnya menjadi *vocabulary dimension*.

   iii) Menghitung probabilitas dari setiap kata di *vocabulary* dengan *softmax*

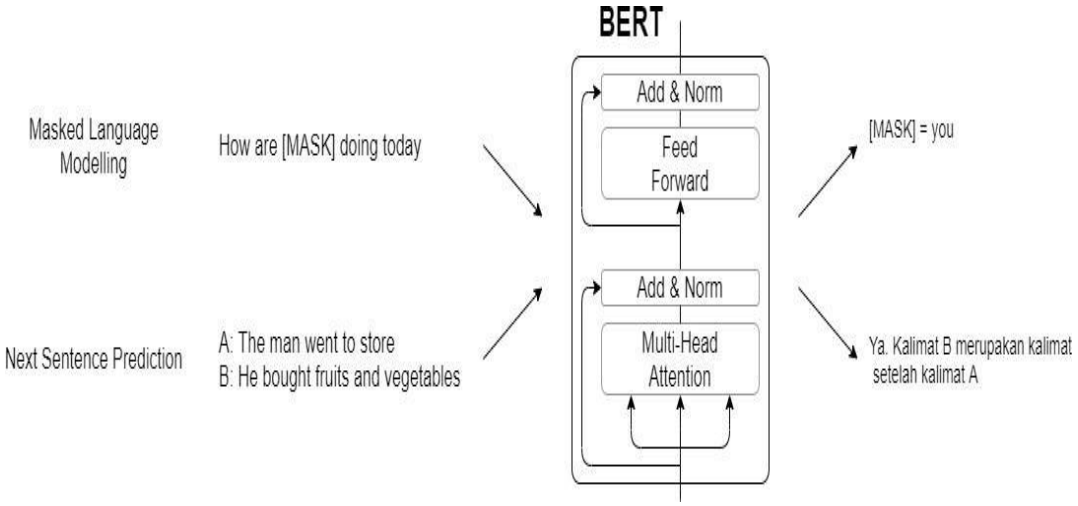



.

**Gambar 2. 8 Proses Masked Language Modelling**

2. *Next Sentence Prediction*

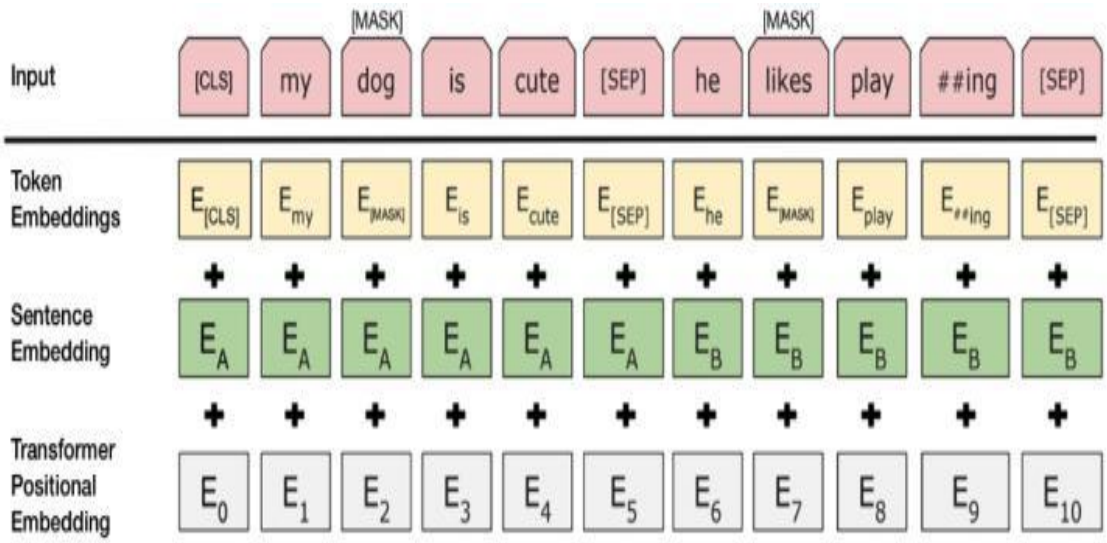
Dalam proses training BERT, model dapat menerima pasangan kalimat sebagai input dan dilatih untuk memprediksi jika kalimat kedua pada pasangan tersebut adalah kalimat berikutnya pada dokumen aslinya atau hanya satu kalimat saja. Selama *training*, 50% dari input adalah pasangan kalimat dimana kalimat kedua adalah kalimat berikutnya pada dokumen asli. Sedangkan 50% lainnya adalah kalimat yang diambil secara acak dari *corpus* sebagai kalimat kedua.




**Gambar 2. 9** Proses Pre-training pada BERT

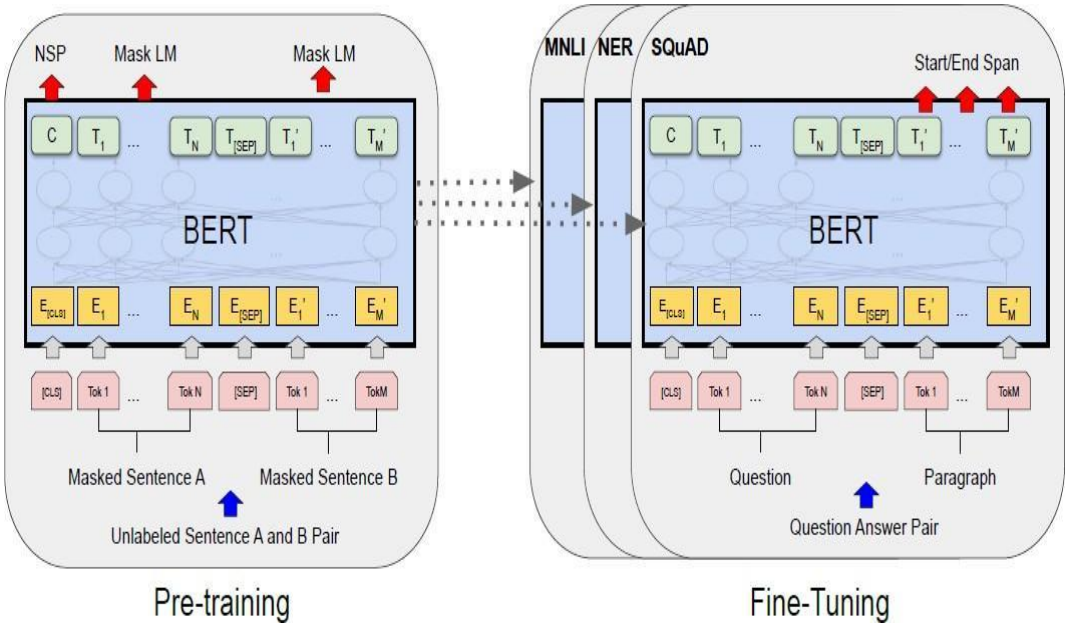
Sebagai representasi input pada BERT, terdapat tiga *embedding layerss* yaitu:

1. *Token embeddings* adalah *layers* pertama yang token masuki, yaitu representasi vektor dari tiap token. Setiap token dalam input akan dipetakan ke representasi vektor berdimensi tinggi dari token yang diberikan. Tiap token diganti menjadi id yang didapatkan berdasarkan *vocabulary*
2. *Sentence embeddings* menunjukkan kalimat pertama atau kalimat kedua, ditambahkan ke setiap token dan digunakan untuk membedakan antar kalimat jika terdapat lebih dari dua kalimat. Lapisan ini hanya memiliki dua representasi: A untuk token yang termasuk dalam kalimat pertama, dan B untuk token yang termasuk dalam kalimat kedua.
3. *Positional embedding* ditambahkan ke setiap token untuk menyimpan informasi tentang posisi kata dalam urutan. Konsep dan implementasi dari *positional embedding* ditunjukkan dalam *Transformers*. BERT telah mempelajari posisi *embedding layers* selama *pre-training*.

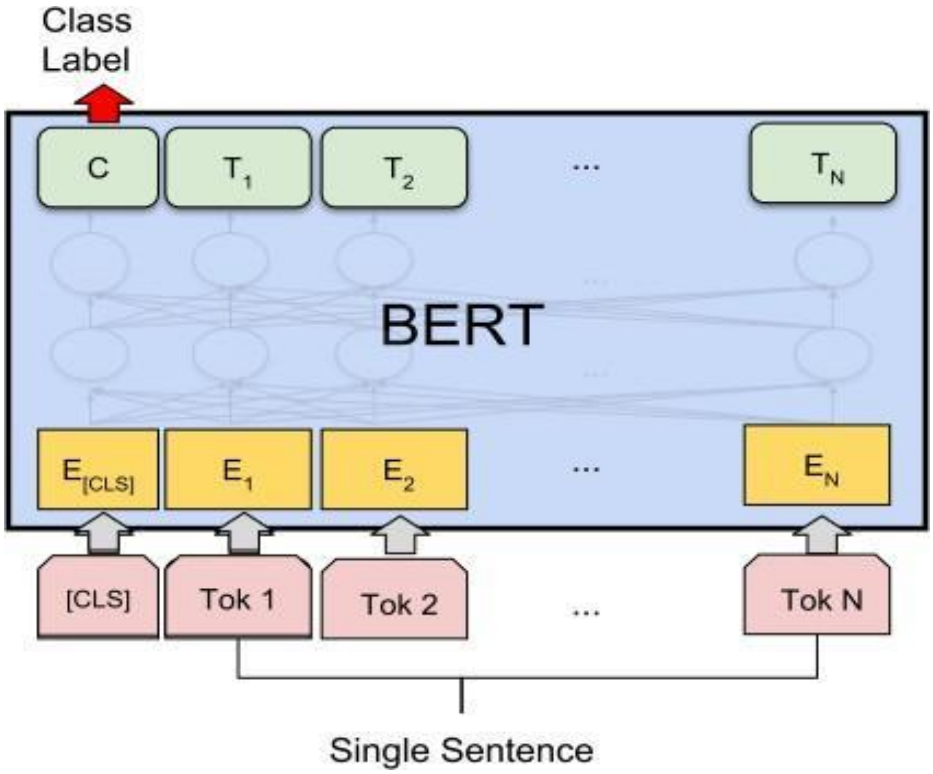



**Gambar 2. 10** Representasi Input pada BERT

Untuk melatih sebuah model bahasa, *classifier* perlu dilatih dengan sedikit perubahan pada model BERT selama fase pelatihan (*training*) yang disebut *finetuning*. Seperti yang dipaparkan oleh Devlin dan rekan-rekannya, terdapat rekomendasi *hyperparameters* yang dapat di-*fine-tuning* untuk mencapai hasil yang maksimal. *Fine-tuning* sangat mudah dilakukan karena mekanisme self-attention di *Transformers* membuat BERT bisa membuat model untuk berbagai tugas, baik pada kalimat tunggal (single sentence) atau kalimat berpasangan, dengan menukar masukan dan keluaran yang sesuai.




**Gambar 2. 11** Prosedur Pre-traning dan Fine-tuning

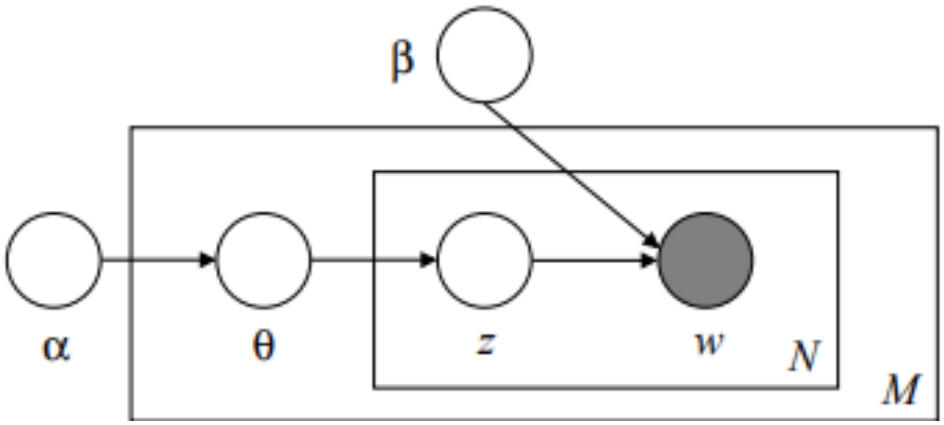



**Gambar 2. 12** Ilustrasi Fine-tuning pada Tugas dengan Single Sentence (Devlin et al., 2019)

## 2.8 Latent Dirichlet Allocation (LDA)

*Latent Dirichlet Allocation* merupakan salah satu metode yang dapat dipilih dalam melakukan analisis untuk dokumen yang memiliki ukuran sangat besar. LDA itu sendiri bisa digunakan untuk meringkas, melakukan klasterisasi, menghubungkan atau memproses data yang sangat besar dikarenakan LDA menghasilkan daftar topik yang diberi bobot untuk masing-masing dokumen. Distribusi yang digunakan yaitu distribusi *Dirichlet,* yang digunakan untuk memperoleh distribusi topik per-dokumen, dalam proses generatif, hasil yang didapatkan dari *Dirichlet* digunakan untuk mengalokasikan kata-kata dalam dokumen untuk topik yang berbeda. Pada LDA, dokumen-dokumen adalah objek yang bisa diamati, namun topik, distribusi topik per-dokumen, penggolongan setiap kata untuk topik per-dokumen adalah struktur tersembunyi. Menurut Blei (2003), LDA merupakan model probabilistik generatif dari kumpulan tulisan yang dapat disebut corpus. Ide dasar dari metode LDA yaitu setiap dokumen direpresentasikan sebagai campuran acak atas topik yang tersembunyi, dimana setiap topik memiliki karakter yang ditentukan berdasarkan distribusi kata-kata yang terdapat didalamnya (Hikmah et al., 2020).

Blei merepresentasikan metode LDA sebagai model probabilistic secara visual seperti pada gambar 2.13 berikut.




**Gambar 2.13** Model Representasi LDA

Dapat dilihat dari Gambar 3.1 diatas yaitu representasi metode LDA menurut (Sahria & Hatta Fudholi, 2017) dimana terdapat tingkatan pada pemodelan dengan LDA. Parameter α dan β yaitu parameter distribusi topik yang berada pada tingkatan corpus, adalah kumpulan dari M dokumen. Untuk parameter α yang digunakan dalam menentukan distribusi topik dokumen, jika nilai alpha semakin besar dalam suatu dokumen, menandakan bahwa campuran topik yang dibahas dalam dokumen semakin banyak. Untuk parameter β yang digunakan untuk menentukan distribusi kata dalam topik. Jika nilai beta semakin tinggi, maka semakin banyak kata-kata yang terdapat di dalam topik, namum jika nilai beta semakin kecil, maka semakin sedikit kata-kata yang terdapat di dalam topik sehingga topik tersebut mengandung kata-kata yang lebih spesifik. pada variabel θm yaitu variabel yang berada di tingkat dokumen (M). Variabel θ merepresentasikan distribusi topik untuk dokumen 21 tersebut. Jika nilai θ semakin tinggi, maka semakin banyak topik yang terdapat di dalam dokumen, jika nilai θ semakin kecil, maka semakin spresifik pada topik tertentu. Pada variabel Zn dan Wn yaitu variabel tingkat kata (N). Variabel Z merepresentasikan topik dari kata tertentu pada sebuah dokumen, pada variabel W merepresentasikan kata yang berkaitan dengan topik tertentu yang terdapat dalam dokumen. Berdasarkan penjelasan notasi sebelumnya, proses generatif pada LDA akan berkorespondensi pada joint distribution dari variabel yang tersembunyi dan variabel yang terobsesi. Berikut merupakan perhitungan probabilitas dari sebuah corpus berdasarkan notasi yang telah dijelaskan (Putra et al., 2023).

$$p(D|a,\beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{Z_{dn}} p(Z_n|\theta) p(W_n|Z_n,\beta)\right) d\theta_d (0,7)$$

$\alpha$ = digunakan dalam menentukan distribusi topik dokumen

$\beta$ = untuk menentukan distribusi kata dalam topik.

$\theta$ = merepresentasikan distribusi topik untuk dokumen tersebut.

$Z$ = merepresentasikan topik dari kata tertentu pada sebuahdokumen

$W$ = merepresentasikan kata yang berkaitan dengan topik tertentu yang terdapat dalam dokumen.

$Z_n$ dan $W_n$ = variabel tingkat kata (N)

Dapat dilihat bahwa pada notasi β mendeskripsikan topik, dimana pada setiap$\beta$ merupakan distribusi dari sejumlah kata. Pada Variabel $\theta_d$ adalah variabel level dokumen dengan satu kali sampel per dokumen yang merepresentasikan proporsi topik untuk dokumen ke *d*. Pada notasi $Z_{dn}$ dan $W_{dn}$ merupakan representasi variabel di level kata dengan satu kali sampel untuk masing-masing kata pada setiap dokumen.

## 2.9 *Topic Coherence*

Topic modeling membahas mengenai kumpulan dari sebuah kata-kata dari sebuah dokumen ataupun corpus. Berdasarkan dari kata-kata yang terdapat dalam dokumen yang digunakan, penggalian dari relasi topik dilakukan dengan asumsi bahwa pada satu dokumen meliputi suatu set kecil dari topik yang ringkas, dimana topik-topik ini perlu dikorelasikan dengan interpretasi manusia. Pada penelitian ini akan menggunakan validasi topik dengan menggunakan coherence topic (Putra, 2017). *Topic Coherence* yaitu dimana satu set dari kata-kata yang dihasilkan pada topik model dengan dinilai berdasarkan tingkat koherensi atau dalam diinterpretasi oleh manusia dengan tingkat kemudahannya. *Topic Coherence* mengukur nilai dari suatu topik dengan mengukur tingkat kesamaan semantik antara kata-kata yang ada dalam topik. Pengkukuran ini dapat membantu dalam membedakan antara topik 22 yang dapat diinterpretasi secara semantik dengan topik yang memiliki keterkaitan secara statistik (Putra, 2017). *Topic Coherence* merupakan suatu ukuran yang akan digunakan untuk mengevaluasi Topic Modeling, dimana jika coherence skor topik yang tinggi maka model yang dihasilkan tersebut yang baik. Menurut Wisdom (2017) *Topic Coherence* dapat dianggap memberikan kemampuan interpretasi lebih baik terhadap hasil dari Topic Modeling dibandingkan dengan Perplexity. Namun hasil dari matriks perplexity terkadang tidak memiliki korelasi yang baik pada interpretasi model oleh manusia (Listari, 2019).