Tugas UTS
Komputasi Intelegensia

Hilmy Rahmadani,
NPM 2206810490
Department Mathematics, University of Indonesia

Abstract

Penelitian ini mengevaluasi performa model BERT dan DistilBERT dalam tugas analisis sentimen pada cuitan terkait kebijakan Pemberlakuan Pembatasan Kegiatan Masyarakat (PPKM) di Indonesia. Model-model ini dinilai berdasarkan metrik accuracy, precision, recall, dan f1-score, serta aspek efisiensi waktu pelatihan dan jumlah parameter yang dapat dilatih. Hasil eksperimen menunjukkan bahwa bahwa BERT memberikan performa evaluasi yang unggul, sementara DistilBERT menawarkan efisiensi waktu pelatihan yang lebih baik. Oleh karena itu, DistilBERT dapat menjadi pilihan yang lebih efisien dalam skenario yang memerlukan pelatihan cepat tanpa penurunan performa yang signifikan.

Keywords: BERT; DistilBERT; Sentimen Analisis

1 Pendahuluan

Kebijakan Pemberlakuan Pembatasan Kegiatan Masyarakat (PPKM) yang diterapkan di Indonesia sejak masa pandemi COVID-19 merupakan upaya pemerintah untuk menekan laju penyebaran virus. Kebijakan ini berdampak langsung pada aktivitas masyarakat di berbagai sektor, termasuk sosial, ekonomi, dan kesehatan. Sebagai langkah strategis, kebijakan PPKM menjadi salah satu topik yang banyak dibicarakan oleh masyarakat, terutama melalui platform media sosial seperti Twitter. Cuitan masyarakat mengenai kebijakan ini menunjukkan berbagai sudut pandang dan sentimen, mulai dari dukungan hingga kritik, yang dapat memberikan wawasan penting bagi pemerintah terkait respons masyarakat.

Analisis sentimen merupakan salah satu metode yang dapat digunakan untuk memahami persepsi publik terhadap suatu topik. Dengan semakin banyaknya data yang tersedia melalui platform media sosial, teknik ini memungkinkan kita untuk memperoleh insight terkait opini masyarakat dalam skala yang luas. Namun, untuk melakukan analisis sentimen dengan akurasi yang baik, diperlukan model pemrosesan bahasa alami (NLP) yang mampu memahami konteks bahasa secara mendalam.

Meskipun demikian, metode tradisional untuk analisis sentimen seringkali terbentur pada keterbatasan pemahaman tentang aspek bahasa dan konteks (Zhang et al., 2022). Misalnya, ulasan negatif dapat berasal dari kesalahpahaman, sementara kata-kata pujian dapat mengandung ironi. Dalam hal ini, pendekatan berbasis BERT (Bidirectional Encoder Representations from Transformers) merupakan inovasi baru. Model bahasa ciptaan yang dikembangkan oleh AI buatan Google bernama BERT yang mampu menganalisis kata dan memahami hubungan antar kata dan konteks kalimat (Alaparthi & Mishra, n.d.; Boukabous & Azizi, 2022; Deepa & Tamilarasi, 2021). Kemampuan inilah yang memungkinkan BERT lebih tepatnya bert-based-caused untuk berfungsi sebagai media untuk mengungkapkan opini publik di balik komentar-komentar pada film. Tujuan penelitian ini adalah bagaimana penerapan metode transformer dengan model bert-based-caused pada komentar-komentar film mampu memprediksi komentar-komentar tersebut apakah bersifat positif atau negatif

Meskipun demikian, BERT memiliki kelemahan, yaitu membutuhkan waktu pelatihan yang lama dan sumber daya komputasi yang besar. Untuk mengatasi keterbatasan tersebut, dikembangkanlah DistilBERT, yaitu versi yang lebih ringan dari BERT yang tetap mempertahankan sebagian besar performa BERT namun dengan waktu pelatihan yang lebih cepat. Penelitian oleh Mahira (2024) menunjukkan efisiensi dan akurasi model DistilBERT pada analisis sentimen pemilihan presiden.

Penelitian ini bertujuan untuk membandingkan performa model BERT dan DistilBERT pada tugas analisis sentimen terkait kebijakan PPKM, serta mengevaluasi efek kuantisasi pada efisiensi model. Diharapkan hasil penelitian ini dapat menjadi acuan dalam memilih model NLP yang efisien dan tetap memiliki performa yang baik untuk analisis sentimen dalam konteks kebijakan publik.

2 Data dan Metode

2.1 Data

Dataset kumpulan cuitan di media sosial X, mengenai opini pembatasan PPKM yang diterapkan di Indonesia pada masa pandemi Covid-19. Dataset ini berasal dari website kaggle yang mencakup 23644 baris dengan 4 kolom. Fitur yang digunakan untuk variabel target adalah "sentiment" karena kita ingin memprediksi sentimen cuitan-cuitan di media sosial X mengenai PPKM. Fitur yang digunakan untuk variabel prediktor adalah "Tweet".

2.2 Metode Penelitian

Teknik klasifikasi yang diterpakan pada penelitian ini di presentasikan pada gambar 1. Digunakan metode Bidirectional Encoder Representations from Transformers (BERT) dan Distilation BERT (DistilBert) untuk menyelesaikan masalah klasifikasi sentimen dengan dataset berbentuk data tabular. Model ini memanfaatkan transformer, sebuah mekanisme attention yang mempelajari hubungan kontekstual antar kata (atau sub-kata) dalam sebuah teks. Dalam bentuk dasarnya, transformer memiliki dua mekanisme terpisah sebuah encoder yang membaca input teks dan sebuah decoder yang menghasilkan prediksi untuk tugas yang diberikan. Berikut adalah penjelasan mengenai kedua metode dan langkah-langkah yang dilakukan sebelum dilakukan pemodelan

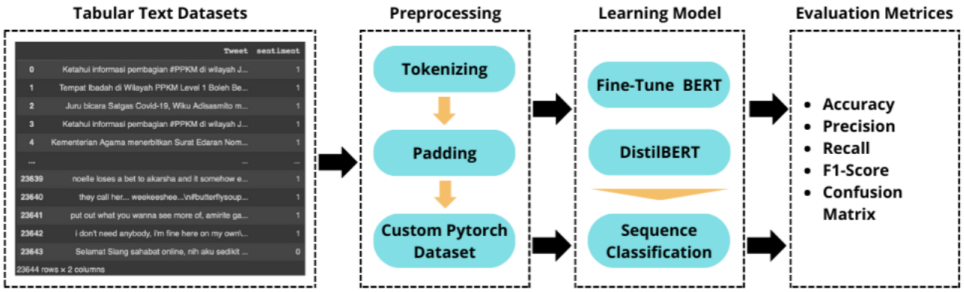

Figure 1: Research Flow Method

2.3 Data Preprocessing

Teknik penghapusan baris yang mengandung Missing values dilakukan agar memastikan semua data memiliki nilai atau input sehingga bisa dilakukan pelatihan model.

2.4 Deep Learning Language Model menggunakan BERT

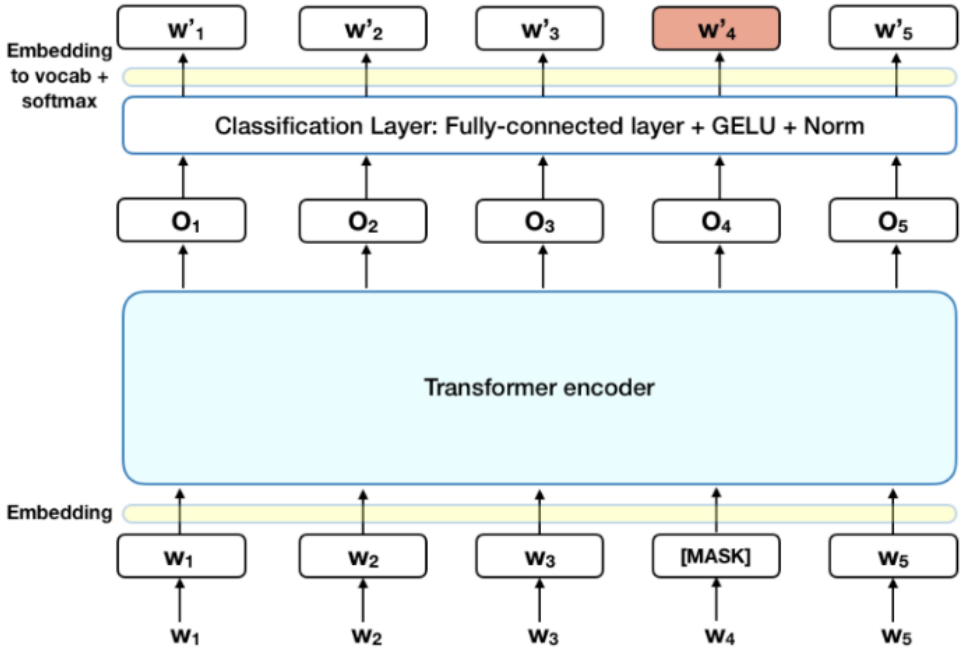

Figure 2: Arsitektur BERT

Model Bidirectional Encoder Representations from Transformers (BERT) adalah model transformasi yang dirancang oleh peneliti google untuk memahami konteks bidirectional dari teks, artinya model ini membaca teks secara menyeluruh dari kiri ke kanan dan dari kanan ke kiri [4]. Arsitektur model BERT di atas dibangun dengan lapisan-lapisan utama seperti berikut:

- **Tokenizer**: Modul ini mengubah teks bahasa menjadi rangkaian bilangan bulat ("token").
- **Embedding**: Modul ini mengonversi urutan token menjadi array vektor bernilai riil yang merepresentasikan token-token tersebut. Modul ini melakukan konversi dari jenis token diskrit menjadi ruang Euclidean berdimensi rendah.
- **Encoder**: Tumpukan blok Transformer dengan *self-attention*, tetapi tanpa *causal masking*. *attention* adalah metode pembelajaran mesin yang menentukan kepentingan relatif dari setiap komponen dalam suatu sekuens atau teks dibandingkan dengan komponen lainnya dalam sekuens atau teks tersebut. Dalam pemrosesan bahasa alami, kepentingan direpresentasikan oleh bobot 'lunak' yang diberikan kepada setiap kata dalam sebuah kalimat [1]. Secara umum, *attention* mengkodekan vektor yang disebut *embedding* token di seluruh urutan dengan lebar tetap yang dapat berkisar dari puluhan hingga jutaan token.

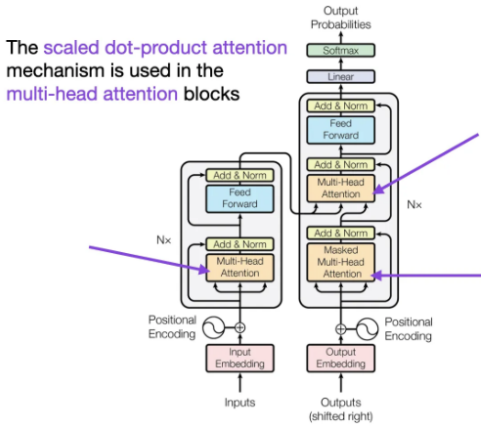

Figure 3: arsitektur lapisan BERT Attention Mechanism

- **Task Head**: Modul ini mengonversi vektor representasi akhir menjadi token yang terenkode satu-hot (one-hot encoded) dengan menghasilkan distribusi probabilitas yang diprediksi di antara jenis token. Modul ini dapat dianggap sebagai decoder sederhana yang menerjemahkan representasi laten menjadi jenis token, atau sebagai "lapisan *un-embedding*".

Task head ini diperlukan saat pre-training, tetapi sering kali tidak dibutuhkan untuk "tugas downstream" seperti *question answering* atau klasifikasi sentimen. Sebagai gantinya, task head ini dihilangkan dan digantikan dengan modul baru yang diinisialisasi sesuai kebutuhan tugas, kemudian modul baru ini di-finetune. Vektor representasi laten dari model langsung dimasukkan ke dalam modul baru ini, memungkinkan *transfer learning* yang efisien dalam menggunakan sampel. gambar 2 memvisualisasikan arsitektur BERT

2.5 Deep Learning Language Model menggunakan DistilBERT

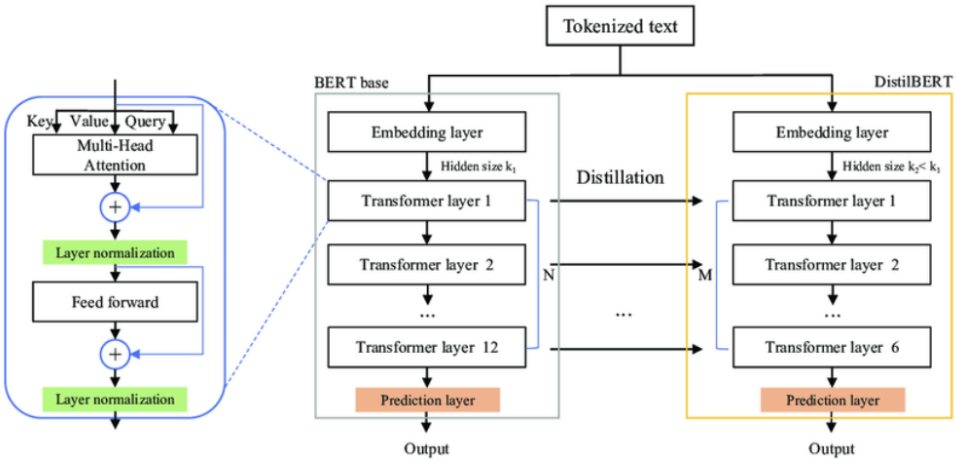

Figure 4: Arsitektur BERT

DistilBERT (Distilled BERT) adalah varian dari model BERT yang disederhanakan yang dikembangkan melalui teknik distilasi pengetahuan (knowledge distillation) untuk mempercepat proses inferensi dan mengurangi kebutuhan sumber daya komputasi, tanpa mengorbankan kinerja secara signifikan. DistilBERT dirancang untuk memahami konteks bidirectional teks seperti BERT namun menggunakan lebih sedikit lapisan dan parameter. DistilBERT mempunyai arsitektur yang sama dengan BERT dengan *token-type emmbedding* dan *pooler* nya dihilangkan sementara banyak lapisannya dikurangi menjadi setengah [7].

2.6 Adaptive Moment Estimation

Optimasi adalah bagian penting dalam pelatihan model *deep learning*, di mana tujuannya adalah untuk meminimalkan nilai fungsi *loss* sehingga model dapat melakukan generalisasi dengan baik. AdamW adalah varian dari algoritma optimasi Adam (*Adaptive Moment Estimation*) yang menambahkan regularisasi dengan metode *weight decay* (pengurangan bobot), yang sangat bermanfaat untuk mencegah *overfitting* pada model yang kompleks seperti BERT [8].

- **Adam Optimizer**: Adam mengkombinasikan dua metode optimasi: *Momentum* dan *RMSProp*. Proses perhitungan parameter pembaruan pada Adam dilakukan dengan dua momen eksponensial dari gradien:
  - Momen pertama (m_t) adalah estimasi rata-rata gradien:

    $$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

    Di mana β_1 adalah faktor *momentum* dan g_t adalah gradien *loss* terhadap bobot.
  - Momen kedua (v_t) adalah estimasi varians gradien:

    $$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

    Di mana β_2 adalah faktor untuk mengendalikan magnitudo perubahan parameter.
- **Bias Correction**: Pada awal pelatihan, momen pertama dan kedua mungkin mengalami bias ke nilai 0. Untuk mengoreksi ini, Adam menerapkan faktor pengoreksi bias:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- **Pembaruan Parameter**: Adam kemudian memperbarui parameter berdasarkan kombinasi dari momen pertama dan kedua yang telah dikoreksi bias:

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

  Di mana α adalah laju pembelajaran dan ϵ adalah nilai kecil untuk mencegah pembagian dengan nol.

- **Weight Decay**: AdamW menambahkan komponen *weight decay* (regulasi L_2) secara eksplisit ke dalam proses optimasi. Ini diterapkan sebagai berikut:

$$\theta_t = \theta_{t-1} - \alpha \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \cdot \theta_{t-1} \right)$$

  Di mana λ adalah koefisien *weight decay*. Penambahan ini mendorong nilai parameter menuju nol, yang membantu mencegah *overfitting*.

2.7 Linear Scheduler with Warmup

Salah satu pilihan paling penting dalam optimasi berbasis gradien adalah *learning rate* (ukuran langkah) η. Jika η terlalu kecil, pembelajaran mungkin berjalan terlalu lambat atau model mungkin terjebak dalam area yang tidak menguntungkan pada lanskap *loss*. Jika η terlalu besar, pelatihan biasanya akan mengalami divergensi. Dalam praktiknya, umum untuk memilih jadwal *learning rate* dinamis η_t. Jadwal *learning rate* modern untuk pembelajaran mendalam biasanya terdiri dari periode *warmup* di mana η_t dinaikkan secara linear dari nol ke nilai target η_{trgt} selama waktu *warmup* T_{wrm}. Setelah periode *warmup*, biasanya *learning rate* akan mengalami penurunan [9].

Linear Warmup: Ini didefinisikan oleh jadwal

$$\eta_t = \eta_{\text{init}} + (\eta_{\text{trgt}} - \eta_{\text{init}}) \left(\frac{t}{T_{\text{wrm}}} \right).$$

Laju *warmup* adalah

$$\alpha := \frac{\eta_{\text{trgt}} - \eta_{\text{init}}}{T_{\text{wrm}}}.$$

$T_{\text{wrm}} = 1$ berhubungan dengan *constant learning rate*. Kecuali jika ditentukan lain, ditetapkan $\eta_{\text{init}} = 0$ saat merujuk ke *linear warmup*.

3 Hasil dan Diskusi

Dalam penelitian ini, eksperimen dilakukan dengan 2 kasus, yaitu kasus pertama dengan model BERT menggunakan 1000 data dengan 80% data pelatihan dengan 10% data pengujian dan 10% data validasi; kasus kedua dengan model BERT dan DistilBERT menggunakan 5000 data dengan 80% data pelatihan dengan 10% data pengujian dan 10% data validasi. Pada tahap awal eksperimen, langkah preprocessing dilakukan dengan tokenisasi data teks, yaitu memecah teks menjadi komponen yang lebih kecil (token) dan dikonversikan menjadi token id yang merepresentasikan kata atau sub-kata tersebut. Selanjutnya dilakukan padding untuk menyamakan panjang semua kalimat agar memiliki panjang yang sama dengan diubah menyesuaikan panjang yang telah ditentukan. Gambar 5 menunjukkan hasil preprocessing data teks dari input awal hingga penerapan padding.

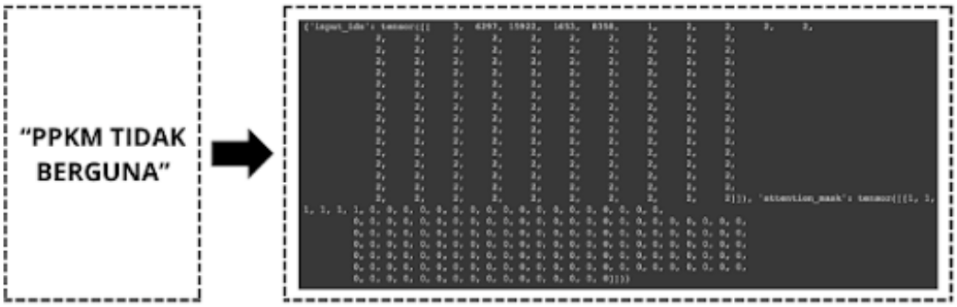

Figure 5: Proses Preprocessing: (kiri) input teks, (kanan) hasil tokenisasi dan padding

Setelah melewati tahap preprocessing, pembelajaran dengan model BERT dan DistilBERT disiapkan seperti pada gambar 6 dan 7. model dilatih untuk total 20 epoch pada kedua kasus. Diterapkan *optimizer* Adam dengan *learning rate* $2e^{-5}$, diterapkan juga *learning rate linear schedule with warmup* dan *loss function cross entropy loss*,

metode ini dipilih karena akan dilakukan klasifikasi multi-kelas. *Batch size* yaitu jumlah sampel yang diproses sebelum model diperbarui dalam pelatihan jaringan neural, diatur menjadi 32.

```
SentimentClassifier(
  (bert): BertForSequenceClassification(
    (bert): BertModel(
      (embeddings): BertEmbeddings(
        (word_embeddings): Embedding(32000, 768, padding_idx=0)
        (position_embeddings): Embedding(512, 768)
        (token_type_embeddings): Embedding(2, 768)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (encoder): BertEncoder(
        (layer): ModuleList(
          (0-11): 12 x BertLayer(
            (attention): BertAttention(
              (self): BertSdpaSelfAttention(
                (query): Linear(in_features=768, out_features=768, bias=True)
                (key): Linear(in_features=768, out_features=768, bias=True)
                (value): Linear(in_features=768, out_features=768, bias=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
              (output): BertSelfOutput(
                (dense): Linear(in_features=768, out_features=768, bias=True)
                (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                (dropout): Dropout(p=0.1, inplace=False)
              )
            )
            (intermediate): BertIntermediate(
              (dense): Linear(in_features=768, out_features=3072, bias=True)
              (intermediate_act_fn): GELUActivation()
            )
            (output): BertOutput(
              (dense): Linear(in_features=3072, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
        )
      )
      (pooler): BertPooler(
        (dense): Linear(in_features=768, out_features=768, bias=True)
        (activation): Tanh()
      )
    )
    (dropout): Dropout(p=0.1, inplace=False)
    (classifier): Linear(in_features=768, out_features=3, bias=True)
  )
  (drop): Dropout(p=0.3, inplace=False)
)
```

```
==========================================================================
Layer (type:depth-idx)                   Param #          Kernel Shape
==========================================================================
├─BertForSequenceClassification: 1-1     --               --
|    └─BertModel: 2-1                    --               --
|    |    └─BertEmbeddings: 3-1          --               --
|    |    |    └─Embedding: 4-1          24,576,000       [768, 32000]
|    |    |    └─Embedding: 4-2          393,216          [768, 512]
|    |    |    └─Embedding: 4-3          1,536            [768, 2]
|    |    |    └─LayerNorm: 4-4          1,536            [768]
|    |    |    └─Dropout: 4-5            --               --
|    |    └─BertEncoder: 3-2             --               --
|    |    |    └─ModuleList: 4-6         85,054,464       --
|    |    └─BertPooler: 3-3              --               --
|    |    |    └─Linear: 4-7             590,592          [768, 768]
|    |    |    └─Tanh: 4-8               --               --
|    └─Dropout: 2-2                      --               --
|    └─Linear: 2-3                       2,307            [768, 3]
├─Dropout: 1-2                           --               --
==========================================================================
Total params: 110,619,651
Trainable params: 110,619,651
Non-trainable params: 0
==========================================================================
```

Figure 6: Desain arsitektur BERTforSequenceClassification

```
DistilSentimentClassifier(
  (bert): DistilBertForSequenceClassification(
    (distilbert): DistilBertModel(
      (embeddings): Embeddings(
        (word_embeddings): Embedding(119547, 768, padding_idx=0)
        (position_embeddings): Embedding(512, 768)
        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (transformer): Transformer(
        (layer): ModuleList(
          (0-5): 6 x TransformerBlock(
            (attention): MultiHeadSelfAttention(
              (dropout): Dropout(p=0.1, inplace=False)
              (q_lin): Linear(in_features=768, out_features=768, bias=True)
              (k_lin): Linear(in_features=768, out_features=768, bias=True)
              (v_lin): Linear(in_features=768, out_features=768, bias=True)
              (out_lin): Linear(in_features=768, out_features=768, bias=True)
            )
            (sa_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (ffn): FFN(
              (dropout): Dropout(p=0.1, inplace=False)
              (lin1): Linear(in_features=768, out_features=3072, bias=True)
              (lin2): Linear(in_features=3072, out_features=768, bias=True)
              (activation): GELUActivation()
            )
            (output_layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          )
        )
      )
    )
    (pre_classifier): Linear(in_features=768, out_features=768, bias=True)
    (classifier): Linear(in_features=768, out_features=3, bias=True)
    (dropout): Dropout(p=0.2, inplace=False)
  )
  (drop): Dropout(p=0.3, inplace=False)
)
```

```
==========================================================================
Layer (type:depth-idx)                   Param #          Kernel Shape
==========================================================================
├─DistilBertForSequenceClassification: 1-1  --            --
|    └─DistilBertModel: 2-1              --               --
|    |    └─Embeddings: 3-1              --               --
|    |    |    └─Embedding: 4-1          91,812,096       [768, 119547]
|    |    |    └─Embedding: 4-2          393,216          [768, 512]
|    |    |    └─LayerNorm: 4-3          1,536            [768]
|    |    |    └─Dropout: 4-4            --               --
|    |    └─Transformer: 3-2             --               --
|    |    |    └─ModuleList: 4-5         42,527,232       --
|    └─Linear: 2-2                       590,592          [768, 768]
|    └─Linear: 2-3                       2,307            [768, 3]
|    └─Dropout: 2-4                      --               --
├─Dropout: 1-2                           --               --
==========================================================================
Total params: 135,326,979
Trainable params: 135,326,979
Non-trainable params: 0
==========================================================================
```

Figure 7: Desain arsitektur DistilBERTforSequenceClassification

Berdasarkan hasil eksperimen, pengukuran hasil setiap kasus dijelaskan dalam Tabel 1 dan Tabel 2. Tabel 1 menampilkan kinerja model arsitektur BERT pada dataset dengan 1000 baris dengan pembagian 800 data pelatihan, 100 data tes, 100 data validasi. Tabel 2 menampilkan kinerja model arsitektur BERT dan DistilBERT pada dataset dengan 5000 baris dengan pembagian 4000 data pelatihan, 500 data tes, 500 data validasi

Metric	Value
Accuracy	95%
Precision	94.53%
Recall	95%
F1-Score	94.71%
Running Time	7 min 15 s
Trainable Params	110619651

Table 1: Kasus 1

Model	Accuracy	Precision	Recall	F1-Score	Running Time	Trainable Params
BERT	95.4%	95.16%	95.4%	95.04%	35 min 40 s	110619651
DistilBERT	93.8%	93.21%	93.8%	93.24%	18 min 45 s	135326979

Table 2: Kasus 2, Perbandingan model BERT dan DistilBERT

Dari Tabel 2, dapat dilihat bahwa arsitektur BERT memiliki performa evaluasi yang paling baik dari segi accuracy, precision, recall, dan f1-score, namun membutuhkan waktu pelatihan yang lebih lama dibandingkan arsitektur DistilBERT. Sementara itu, arsitektur DistilBERT menawarkan waktu pelatihan yang lebih cepat dengan sedikit penurunan performa evaluasi dibandingkan BERT. Oleh karena itu, DistilBERT menjadi pilihan yang lebih efisien dan cepat dalam proses pelatihan model.

4 Kesimpulan

Studi ini mengkonfirmasi bahwa fine-tuning model BERT dan DistilBERT pada dataset klasifikasi sentimen menghasilkan akurasi yang tinggi, sehingga cocok untuk aplikasi praktis. BERT mencapai akurasi yang lebih tinggi, sementara DistilBERT memberikan akurasi yang hampir sebanding dengan kebutuhan komputasi yang lebih rendah dan waktu pelatihan yang lebih cepat, sehingga lebih layak untuk lingkungan yang membutuhkan pemrosesan real-time atau dengan sumber daya terbatas. Penelitian selanjutnya dapat mengeksplorasi teknik optimasi lebih lanjut seperti kuantisasi dan pruning untuk meningkatkan efisiensi pada kasus sentimen analisis.

5 Referensi

1 H. Singh, *BERTasticity — Understanding Transformers, the CORE behind the Mammoth (Bert)*, Medium, Nov 26, 2019. Available: https://medium.com/@himanshuit3036/bertasticity-part-1-639c9101bb9e

2 R. O. Reboul, *Distillation of BERT-like Models: The Theory*, Towards Data Science, Dec 10, 2021. Available: https://towardsdatascience.com/distillation-of-bert-like-models-the-theory-32e19a02641f

3 H. Adel, A. Dahou, A. Mabrouk, M. E. Abd Elaziz, M. Kayed, I. El-henawy, S. Alshathri, A. Ali, *Improving Crisis Events Detection Using DistilBERT with Hunger Games Search Algorithm*, Mathematics, vol. 10, no. 447, 2022. doi: 10.3390/math10030447.

4 R. Shaikh, *A Comprehensive Guide to Understanding BERT: From Beginners to Advanced*, Medium, Aug 26, 2023. Available: https://medium.com/@shaikhrayyan123/a-comprehensive-guide-to-understanding-bert-from-beginners-to-advanced-2379699e2b51

5 A. Aljabar, B. M. Karomah, *Mengungkap Opini Publik: Pendekatan BERT-based-caused untuk Analisis Sentimen pada Komentar Film*, Journal of System and Computer Engineering (JSCE), vol. 5, no. 1, pp. 1-10, Jan. 2024. Available: https://journal.unpacti.ac.id/index.php/JSCE/article/view/1060/629

6 M. Putri, *Studi Empiris Model BERT dan DistilBERT: Analisis Sentimen pada Pemilihan Presiden Indonesia*, UIN Jakarta, 2024. Available: https://repository.uinjkt.ac.id/dspace/bitstream/123456789/77084/1/MAHIRA%20PUTRI-FST.pdf

7 V. Sanh, L. Debut, J. Chaumond, T. Wolf, *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*, arXiv, 2019. Available: https://arxiv.org/abs/1910.01108

8 D. P. Kingma, J. L. Ba, *ADAM: A Method for Stochastic Optimization*, arXiv, 2014. Available: https://arxiv.org/pdf/1412.6980

9 D. S. Kalra, M. Barkeshli, *Why Warmup the Learning Rate? Underlying Mechanisms and Improvements*, arXiv, 2024. Available: https://arxiv.org/pdf/2406.09405