KERJA PRAKTIK - EF234603

Implementasi Model Bi-LSTM Menggunakan Representasi Vektor Kata Berbasis BERT yang Diintegrasi dengan Informasi Kata-Kelas untuk Klasifikasi Teks

Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo Surabaya 60111
Periode: 7 Oktober 2024 - 23 Desember 2024

Oleh:

Sandhika Surya Ardyanto 5025211022

Pembimbing Departemen
Imam Mustafa Kamal, S.ST, Ph.D

Pembimbing Lapangan
Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

KERJA PRAKTIK – EF234603

Implementasi Model Bi-LSTM Menggunakan Representasi Vektor Kata Berbasis BERT yang Diintegrasi dengan Informasi Kata-Kelas untuk Klasifikasi Teks

Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo Surabaya 60111
Periode: 7 Oktober 2024 - 23 Desember 2024

Oleh:

Sandhika Surya Ardyanto 5025211022

Pembimbing Departemen
Imam Mustafa Kamal, S.ST, Ph.D

Pembimbing Lapangan
Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2025

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR PSEUDOCODE

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN
KERJA PRAKTIK

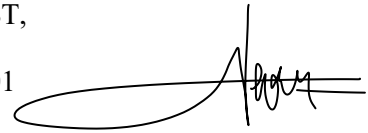
Implementasi Model Bi-LSTM Menggunakan Representasi Vektor Kata Berbasis BERT yang Diintegrasi dengan Informasi Kata-Kelas untuk Klasifikasi Teks

Oleh:

Sandhika Surya Ardyanto 5025211022

Disetujui oleh Pembimbing Kerja Praktik:

1. Imam Mustafa Kamal, S.ST, Ph.D
   NIP. 199203122024061001



(Pembimbing Departemen)

2. Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
   NIP. 197512202001122002

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Implementasi Model Bi-LSTM Menggunakan Representasi Vektor Kata Berbasis BERT yang Diintegrasi dengan Informasi Kata-Kelas untuk Klasifikasi Teks

Nama Mahasiswa	: Sandhika Surya Ardyanto
NRP	: 5025211022
Departemen	: Teknik Informatika FTEIC-ITS
Pembimbing Departemen	: Imam Mustafa Kamal, S.ST, Ph.D
Pembimbing Lapangan	: Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.

ABSTRAK

Klasifikasi teks berita merupakan salah satu topik pada Natural Languange Processing (NLP) yang sering berkaitan dengan analisis sentimen, pendeteksian berita palsu, pendeteksian depresi, dan sistem rekomendasi. Tingkat kepercayaan masyarakat akan ditentukan besarnya kebenaran berita yang beredar sehingga penting untuk membedakan apakah berita yang didapat termasuk berita palsu atau tidak. Representasi vektor kata atau word embedding merupakan metode merepresentasikan kata atau teks menjadi vektor numerik yang dapat dioperasikan oleh mesin untuk memahami konteks kata. Penggunaan model deep learning Bi-LSTM sebagai model klasifikasi teks dan juga transformer BERT untuk menangkap semantik dan sintatik antar kata melalui word embedding.

Percobaan menunjukkan model cukup optimal untuk klasifikasi 2 label dengan akurasi terendah 58,55% dan tertinggi 66,98% sedangkan klasifikasi 2 label masih dibawah 30%.

Kata Kunci : Klasifikasi Teks, NLP, Bi-LSTM, BERT, Transfer Learning, Word Embedding

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT atas penyertaan dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik yang berjudul: Implementasi Model Bi-LSTM Menggunakan Representasi Vektor Kata Berbasis BERT yang Diintegrasi dengan Informasi Kata-Kelas untuk Klasifikasi Teks.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Bapak Imam Mustafa Kamal, S.ST, Ph.D selaku dosen pembimbing kerja praktik.
3. Ibu Prof. Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku pembimbing lapangan selama kerja praktik berlangsung.
4. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 5 Juli 2025
Sandhika Surya Ardyanto

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

1.1. Latar Belakang

Internalisasi dari pengetahuan melalui berita yang beredar di masyarakat akan membentuk perilaku sosial. Jika masyarakat sering menerima berita hoax maka masyarakat akan cenderung mempercayai berita tanpa memvalidasi berita terlebih dahulu[1]. Dengan berkembangnya teknologi, memudahkan penyebaran berita yang ditandai dengan meningkatnya aktivitas media sosial secara drastis[2].

Klasifikasi teks merupakan pendekatan NLP (Natural Languange Processing) yang dapat menjadi cara untuk membedakan apakah berita tersebut termasuk hoax atau tidak. Klasifikasi teks dapat juga meliputi analisis sentimen dan sistem rekomendasi topik melalui word embedding. Model transformer melakukan word embedding untuk memahami hubungan semantik antar kata dengan mengubahnya menjadi representasi vektor numerik. Transformer yang akan digunakan adalah BERT (Bidirectional Encoder Representations from Transformers) memiliki attention mechanism yang memungkinkan memahami konteks kata secara global karena dapat memproses semua kata disekitarnya secara paralel[3]. Token dari BERT akan berupa representasi kata yang kaya konteks untuk kalimat yang panjang dan kompleks sehingga membutuhkan model untuk menagkap representasi kata tersebut[4]. Model Bi-LSTM (Bidirectional Long Short Term Memory) dapat memproses input 2 arah secara sequential sehingga efektif dalam memproses hasil embedding dalam menangkap hubungan kontekstual dari sejumlah token dari BERT yang tersebar cukup jauh. Kedua model dibutuhkan untuk meningkatkan akurasi model dalam memproses urutan kata secara sequential dengan tepat menginterpretasikan konteks kata secara global.

Latar belakang tersebut merupakan kontribusi kerja praktik dengan melakukan implementasi model Bi-LSTM menggunakan representasi vektor kata berbasis BERT yang diintegrasi dengan informasi kata-kelas untuk klasifikasi teks

1.2. Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik sebesar 4 sks dan memberikan kontribusi untuk penelitian mahasiswa S2 dalam implementasi model Bi-LSTM menggunakan representasi vektor kata berbasis BERT yang diintegrasi dengan informasi kata-kelas untuk klasifikasi teks.

1.3. Manfaat

Manfaat yang diperoleh dari implementasi model klasifikasi teks berita antara lain adalah menghasilkan model yang dapat memudahkan pembaca untuk membedakan berita yang dapat dipercaya dan yang tidak.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana melakukan preprocessing pada implementasi model Bi-LSTM menggunakan representasi vektor kata berbasis BERT yang diintegrasi dengan informasi kata-kelas untuk klasifikasi teks ?
2. Bagaimana performa model Bi-LSTM untuk klasifkasi berita dari representasi vektor kata berbasis BERT?
3. Bagaimana pengaruh melatih semua encoder layer dari BERT untuk keakuratan klasifikasi model Bi-LSTM?
4. Bagaimana pengaruh learning terhadap akurasi klasifkasi model Bi-LSTM?

1.5. Lokasi dan Waktu Kerja Praktik

Pengerjaan kerja praktik ini lakukan secara hybrid. Adapun kerja praktik dimulai pada tanggal 7 Oktober 2024 sampai dengan 23 Desember 2024.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Untuk mengetahui kebutuhan dari implementasi model klasifikasi teks berita dari representasi vektor kata berbasis BERT, saya mengikuti rapat dengan Prof Chastine selaku pembimbing lapangan dan. Dalam rapat tersebut, Prof Chastine menyampaikan hasil pertemuan beliau dengan Pak Arya (Mahasiswa S2 yang sedang melakukan thesis). Pada saat rapat, saya mendapatkan pemahaman bagaimana konsep klasifikasi teks berita dari representasi vektor berbasis word embedding bekerja. Selain itu, juga terdapat sejumlah skenario yang diperlukan selama pengerjaan implementasi.

1.6.2. Studi Literatur

Setelah mendapat gambaran bagaimana model klasifikasi tersebut akan digunakan, saya diberitahu model word embedding apa yang akan diimplementasikan untuk menghasilkan representasi vektor kata. Arsitektur model yang diimplementasikan meliputi layer Bi-LSTM dan juga model transfer learning BERT. Selain itu, saya diberikan dataset yang digunakan dalam pengerjaan implementasi tersebut.

1.6.3. Analisis dan Perancangan Sistem

Setelah mengetahui arsitektur model yang dipaparkan, saya melakukan analisa bagaimana model tersebut bekerja. Setelah cukup memahami arsitektur model, saya merancang model dengan berbagai parameter.

1.6.4. Implementasi Sistem

Implementasi merupakan realisasi dari tahap perancangan. Pada tahap ini saya melakukan *hyperparameter tuning* pada model klasifikasi tersebut.

1.6.5. Pengujian dan Evaluasi

Setelah melakukan *hyperparameter tuning*, evaluasi terhadap model diperlukan untuk menilai keakuratan model klasfikasi. Data testing digunakan untuk mengetahui secara kuantitatif performa model dalam mengklasifikasikan.

1.6.6. Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Instansi

Bab ini berisi sejarah instansi, visi instansi, misi instansi, struktur instansi, laboratorium, dan lokasi instansi.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Analisis dan Perancangan Infrastruktur Sistem

Bab ini berisi uraian perancangan yang dilakukan untuk proses implementasi model klasifikasi.

1.7.5. Bab V Implementasi Sistem

Bab ini berisi implementasi yang dilakukan untuk proses implementasi model klasifikasi.

1.7.6. Bab VI Pengujian dan Evaluasi

Bab ini berisi hasil evaluasi dan pembahasan dari model klasifikasi yang telah dikembangkan selama pelaksanaan kerja praktik.

1.7.7. Bab VII Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II
PROFIL INSTANSI

2.1. Sejarah Instansi

Pada masa sekarang pun invasi teknologi informasi sudah dapat dirasakan di berbagai aspek kehidupan manusia. Hal tersebut dapat menjadi tulang punggung pertumbuhan ekonomi negara, baik masa sekarang maupun masa depan. Kesadaran akan hal tersebut, sejak Repelita V yang lalu pemerintah memutuskan untuk memprioritaskan pendidikan tinggi di bidang komputer dan informatika bersama dengan sejumlah disiplin ilmu lain seperti rekayasa, perilaku, manajemen, akuntansi, dan kesenian. Dalam rangka mempersiapkan bangsa Indonesia dalam menghadapi era pembangunan industri dan informasi, pemerintah melalui Direktorat Jendral Pendidikan Tinggi pada tahun 1985 menginstruksikan pendidikan tinggi untuk membuka Program Studi S1 baru di bidang ilmu teknologi komputer di empat universitas atau institut yang terdapat program ITS. Sejak tahun 1993, program ITS yang awalnya disebut Program Studi Teknik Komputer diubah menjadi Jurusan Teknik Komputer. Pada tanggal 11 Juli 1996 akhirnya jurusan ini berganti nama menjadi Jurusan Teknik Informatika berdasarkan Surat Keputusan Direktur Jendral Pendidikan Tinggi Nomor 224/DIKTI/Kep/1996. Dengan keluarnya Surat Keputusan Badan Akreditasi Nasional Perguruan Tinggi (BAN-PT) Nomor 003/BAN-PT/Ak-X/S1/V/2006 pada tanggal 18 Mei 2006, Jurusan Teknik Informatika mendapat nilai akreditasi A. Tidak hanya program Sarjana (S1), jurusan ini juga menawarkan program Pasca Sarjana (S2) sejak tahun 1994 berdasarkan surat keputusan Direktur Jendral Pendidikan Tinggi No.

2851/D/T/2001 berkaitan dengan ijin penyelenggaraan Program-Program Studi Jenjang Program Strata-2 (S2) pada Institut Teknologi Sepuluh Nopember Surabaya. Jurusan Teknik Informatika juga mulai menyelenggarakan program Doktor (S3) sejak tahun 2011. [5]

2.2. Visi Instansi

Sejalan dengan visi ITS untuk menjadi perguruan tinggi dengan reputasi internasional pada bidang ilmu pengetahuan, teknologi, dan seni, yang berfokus untuk menunjang industri dan kelautan yang berwawasan lingkungan. Visi Departemen Informatika adalah menjadi inovator bidang informatika yang unggul di tingkat nasional dengan reputasi internasional, serta berperan aktif dalam upaya memajukan dan mensejahterakan bangsa. Visi Departemen Teknik Informatika untuk menjadi lembaga pendidikan dan penelitian dalam informatika cerdas, mendukung transformasi digital dan berkontribusi kepada masyarakat dengan reputasi internasional. [6]

2.3. Misi Instansi

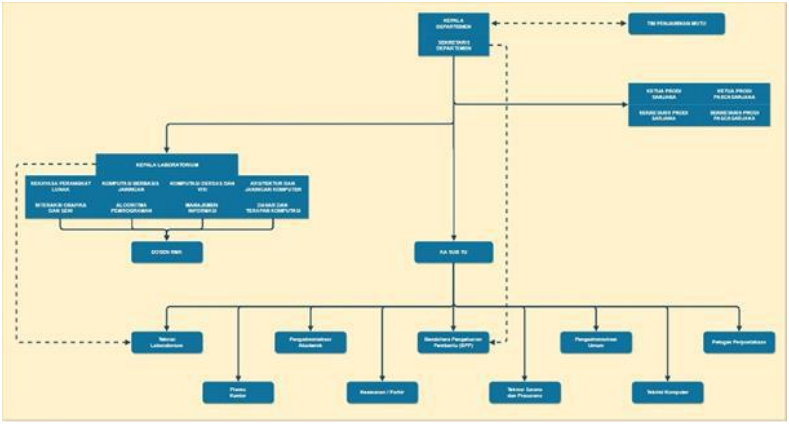
Misi-misi yang dimiliki oleh Departemen Teknik Informatika, ITS adalah sebagai berikut.

1. Menyelenggarakan proses pembelajaran yang berkualitas, dan memenuhi standar nasional maupun internasional.
2. Melaksanakan penelitian yang inovatif, bermutu, dan bermanfaat.
3. Meningkatkan pemanfaatan teknologi informasi dan komunikasi untuk masyarakat.

4. Menjalin kemitraan dengan berbagai lembaga, baik di dalam maupun di luar negeri. [6]

2.4. Struktur Instansi

Gambar 2.1 merupakan gambar struktur instansi yang terdapat di Departemen Teknik Informatika, ITS



Gambar 2. 1 Struktur Instansi Departemen Teknik Informatika, ITS [7]

2.5. Laboratorium

Mahasiswa Departemen Teknik Informatika ITS memiliki akses ke fasilitas laboratorium yang terdiri dari laboratorium bidang minat dan laboratorium workshop. Jenis laboratorium bidang minat mencakup berbagai mata kuliah yang dapat dipelajari oleh mahasiswa Departemen Teknik Informatika, sebagai berikut: [8]

1. Laboratorium Rekayasa Perangkat Lunak

   Keahlian untuk menguji perangkat lunak, mengelola proyek perangkat lunak, mengurangi resiko kesalahan

perangkat lunak, dan membuat perangkat lunak game adalah bidang minat di laboratorium ini.

2. Laboratorium Komputasi Cerdas dan Visi

   Kemampuan lulusan dalam memanipulasi dan menganalisis data citra dalam berbagai bidang aplikasi (AI, biomedika, industri), menerapkan metode sistem cerdas dalam berbagai bidang aplikasi, memodelkan dan mengoptimalkan 9 sistem nyata adalah bidang keahlian yang ditekankan di laboratorium ini.

3. Laboratorium Grafika, Interaksi, dan Game

   Di bidang minat ini, laboratorium menawarkan keahlian yang berfokus pada kemampuan lulusan dalam mendesain, mengembangkan, mencatat proses pembuatan game yang standar, serta menggunakan game engine untuk membuat model tiga dimensi, pemrograman, dan aplikasi realitas virtual tiga dimensi.

4. Laboratorium Manajemen Cerdas Informasi

   Laboratorium di bidang minat ini menekankan keahlian lulusan dalam menganalisis, mensintesis, dan mengevaluasi proses bisnis dan sistem informasi, memasukkan rekayasa pengetahuan ke dalam aplikasi, melakukan investigasi, pengujian, dan evaluasi kematangan dan kepatutan prosedur dan tata kelola teknologi informasi.

5. Laboratorium Komputasi Berbasis Jaringan

   Laboratorium yang menekankan kemampuan sarjana, magister, dan doktor dalam membangun infrastruktur jaringan

yang aman, sistem grid, aplikasi standar jaringan, dan aplikasi multimedia berbasis jaringan.

6. Laboratorium Teknologi Jaringan dan Keamanan Siber Cerdas

   Laboratorium bidang minat ini menawarkan bidang keahlian yang menekankan kemampuan lulusan dalam menerapkan keamanan jaringan dan berbagai arsitektur jaringan yang sesuai dengan standar teknologi terbaru.

7. Laboratorium Algoritma dan Pemrograman

   Kemampuan untuk merancang dan menganalisa algoritma berdasarkan kaidah pemrograman yang kuat dengan tujuan menyelesaikan masalah secara efektif dan efisien, kemampuan untuk menggunakan model pemrograman yang ada mendasari berbagai bahasa pemrograman yang ada, kemampuan untuk memilih bahasa pemrograman untuk menghasilkan aplikasi yang sesuai, seperti mengembangkan sistem atau aplikasi berbasis kerangka kerja, dan peran yang sesuai dengan bidang keahlian yang ditekankan di laboratorium ini

8. Laboratorium Pemodelan dan Komputasi Terapan

   Pemodelan dan simulasi, peramalan sains, optimasi, dan komputasional saintifik adalah semua bidang di mana laboratorium ini menerima penelitian dan kerja sama industri.

Laboratorium workshop yang disediakan oleh Departemen Teknik Informatika antara lain :

1. Workshop Pemrograman 1

   Laboratorium ini memiliki fasilitas unggulan dalam hal teknologi manajemen komputer dan fasilitas fisik

representatif (PC, sistem suara video, jaringan). Selain itu, ruangannya dapat menampung sebanyak 54 orang.

2. Workshop Pemrograman 2

Laboratorium ini memiliki fasilitas unggulan dalam hal teknologi manajemen komputer dan fasilitas fisik representatif (PC, sistem suara video, jaringan). Selain itu, ruangannya dapat menampung sebanyak 54 orang.

3. Laboratorium Pascasarjana S2

Laboratorium ini memungkinkan mahasiswa program master untuk menyelesaikan tugas kuliah dan tesis, seperti studi literatur, uji coba aplikasi dan data, dan penulisan tesis. Mereka juga dapat menggunakan laboratorium bidang minat mereka sendiri.

4. Laboratorium Pascasarjana S3

Mahasiswa program doktor dapat menggunakan laboratorium ini untuk menyelesaikan tugas kuliah dan disertasi, seperti studi literatur, uji coba aplikasi dan data, dan penulisan disertasi. Laboratorium Pascasarjana S3 terdiri dari tiga lantai. Laboratorium S3 di lantai 1 digunakan oleh mahasiswa S3 yang belum mendaftar, Laboratorium Pascasarjana di lantai 3 digunakan oleh mahasiswa S3 yang sudah mendaftar, dan Laboratorium Kerjasama di lantai 3 digunakan oleh mahasiswa S3 yang bekerja sama. Mahasiswa program doktor juga dapat menggunakan laboratorium bidang minat mereka sesuai dengan bidang penelitian mereka.

2.6. Lokasi

Departemen Teknik Informatika, Kampus ITS Sukolilo, Surabaya 60111.

[Halaman ini sengaja dikosongkan]

BAB III
TINJAUAN PUSTAKA

3.1. Penelitian Terkait

Penelitian [9] pada tahun 2024 untuk mendeteksi berita hoax terutama di media sosial dengan akurat. Hal ini dilakukan mengingat autentikasi berita membutuhkan waktu dan biaya yang cukup besar. Pendekatan yang diambil dengan menggabungkan BERT dan Bi-LSTM untuk melakukan klasifikasi berita menjadi benar atau palsu secara otomatis dan efisien. Model BERT yang digunakan adalah bert-base-uncased, yang terdiri dari 12 lapisan encoder, 768 dimensi vektor per token, dan 12 head atensi, dengan total sekitar 110 juta parameter. Sebelum proses training, penulis menghapus data duplikat dan mengubah data menjadi bertipe data *string* kemudian melakukan preprocessing yang dibutuhkan untuk model klasik seperti Word2Vec dan GloVe dengan menghilangkan *stopwords* dan mengubah teks menjadi *lowercase*. Langkah *preprocessing* selanjutnya adalah *tokenisasi* dimana proses pemecahan kata menjadi sejumlah *token* yang nantinya dikonversi menjadi ID numerik untuk diproses model. Setelah *tokenisasi*, proses *lematisasi* dimana *token* yang sudah ada dikonversi ke bentuk dasar atau *lema*-nya. Sebelum menyimpan data sebagai file CSV penulis menghilangkan sejumlah record yang mempunyai *missing values*. Hasil dari model BERT dan Bi-LSTM kemudian menghasilkan probabilitas apakah terrmasuk real news atau fake news dengan fungsi aktivasi *sigmoid*. Terdapat 3 dataset yang digunakan dalam penelitian ini yaitu FakeNewsNet - PolitiFact , GossipCop, dan ISOT. Terdapat model yang diuji yaitu hanya menggunakan BERT, BERT dengan LSTM, dan BERT dengan Bi-LSTM. Berdasarkan evaluasi confusion matrix BERT selalu lebih

unggul untuk dataset FakeNewsNet – PolitiFact dengan akurasi 89.70% dan GossipCop dengan akurasi 85.50%. Model BERT kurang unggul untuk dataset ISOT dimana akurasi dari model BERT + LSTM sebesar 95.92% sedangkan BERT + Bi-LSTM sebesar 95.45%[9].

Penelitian lain pada tahun 2024 [10] menggunakan RoBERTa (*Robustly Optimized BERT Approach*) yang merupakan versi model transformer dengan basis model BERT yang ditingkatkan[10]. Berdasarkan penelitian [11] oleh tim Facebook AI yang memperkenalkan RoBERTa, kedua model BERT dan RoBERTa menggunakan MLM (*Masked Languange Model*) dimana BERT menggunakan MLM statis sedangkan RoBERTa menggunakan MLM dinamis. Perbedaan MLM dinamis terletak pada posisi masking pada *token* yang selalu berubah selama iterasi *epoch* training sedangkan MLM statis terbatas pada variasi *masking* yang ditentukan sebelum proses training sehingga variasi akan selalu sama untuk setiap *epoch*. Selain dari segi MLM, RoBERTa juga menghilangkan NSP *(Next Sentence Prediction*) yang kurang berdampak signifikan dan berpotensi menambah noise saat training sedangkan berdasarkan evaluasi dengan menggunakan metode *FULL-SENTENCE* dan *DOC-SENTENCE* selalu setara atau lebih baik dari NSP. Metode *FULL-SENTENCE* mempelajari konteks hubungan antar kalimat pada blok panjang 512 token alih-alih menebak kebenaran urutan kata. *FULL-SENTENCE* bekerja dengan memenuhi 512 token dari kalimat pada dokumen dan memisahkan dengan token separator jika berbeda sedangkan *DOCS-SENTENCE* menghentikan pengambilan token jika kalimat berakhir pada dokumen meskipun belum 512 token. Selain itu, *corpora* yang digunakan kurang lebih 10 kali lebih banyak dari BERT yang hanya 16 GB. Hal ini memberikan model RoBERTa cakupan bahasa yang lebih luas dan

kontekstual. Berdasarkan penelitian [10], kedua model BERT dan RoBERTa menggunakan 3 jenis *tokenizer* yaitu BERT *Tokenizer*, RoBERTa *Tokenizer*, dan GPT-2 *Tokenizer*. BERT *Tokenizer* menggunakan *WordPiece* yang memecah *word* menjadi *sub-word* berdasarkan probabilitas maksimum dari pasangan sub-word sedangkan GPT-2 Tokenizer dan RoBERTa *Tokenizer* menggunakan BPE (*Byte Pair Encoding*) yang membuat pasangan *token* berdasarkan frekuensi paling sering muncul sebuah token sehingga tokenizer dengan WordPiece memiliki akurasi prediksi kontekstual lebih tinggi meskipun dengan trade off waktu training lebih lama. Pada Liar dataset, PolitiFact.com terdapat 6 label dimana label pants-of-fire, false, dan barelytrue diklasifikasi sebagai fake messages dan label half-true, mostly-true, dan true diklasifikasikan sebagai true messages. Tahap *preprocessing* dilakukan dengan *cleaning text* dengan menghapus IP Address dan URL yang tidak relevan, mengganti *string* kosong dengan "none", dan menghapus *stopwords*. Kemudian memperbaiki ejaan dan *stemming* agar dapat lebih mudah menangkap makna kontekstual. Terakhir, proses *tokenisasi* dan penggabungan *token* menjadi teks bersih yang siap digunakan sebagai input ke dalam model *transformer*. Setelah training model *transformer* didapat hasil akurasi RoBERTa lebih tinggi daripada BERT dengan *tokenizer* masing-masing dan dengan menggunakan GPT-2 *Tokenizer* selalu mendapat akurasi lebih rendah karena kurang optimal dengan arsitektur dan *pretraining* model tersebut

3.2. Word Embedding

Klasifikasi berita mengharuskan model dapat memiliki pemahaman konstektual dari berita yang diberikan dimana

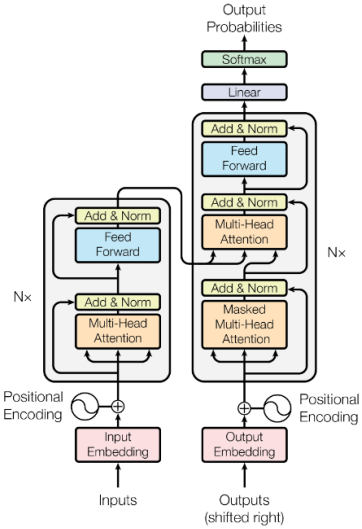
merupakan topik utama dari NLP. Selain dari *word embedding*, cara lain yang lebih klasik adalah menggunakan *one-hot encoding* dan BOW (*Bag of Words*)[12]. Kelemahan *one-hot encoding* adalah membuat representasi vektor kata sejumlah kata unik dari dataset sehingga memori menjadi sangat besar dan juga tidak terdapat informasi dekatnya makna semantik antar kata pada representasi vektor. Kelemahan BOW adalah hanya merepresentasikan frekuensi kemunculan setiap kata yang tidak memiliki informasi urutan kata meskipun jumlah representasi vektor lebih hemat dari *one-hot encoding*. Kelemahan utama keduanya adalah tidak mempertimbangkan makna semantik yang mana menjadi kegunaan utama dari *word embedding*. *Word embedding* merupakan pemetaan representasi yang berisi vektor bilangan real ke ruang n vektor dimensional dimana semakin dekat arti semantik antar kata akan memiliki nilai vektor yang berdekatan pada ruang vektor. Dengan jarak vektor yang berdekatan tersebut mesin atau model dapat seperti memiliki pemahaman semantik seperti manusia.

Model yang menggunakan word embedding seperti Word2Vec[13], FastText[14], dan GloVe[15]. Model Word2Vec merupakan metode berbasis neural networks dimana terdapat 2 model didalamnya yaitu CBOW (*Continuous Bag Of Words*) dan *Skip-gram*. CBOW menerima input konteks kemudian memprediksi kata dalam suatu kalimat berdasarkan kata disekitarnya sedangkan *Skip-gram* menebak konteks dari input kata target. FastText merupakan pengembangan dari Word2Vec yang menghasilkan *word embedding* pada tingkat kata sedangkan FastText dapat membuat kata menjadi sejumlah n-gram atau karakter sehingga dapat memberi informasi semantik meskipun kata tersebut belum ada dalam model *pre-trained* atau OOV (*Out of Vocabulary*). Model GloVe merupakan *word matrix based methods* dimana menggunakan *global matrix factorization* yang melatih representasi kata agar hasil dot product antara dua vektor kata mendekati logaritma frekuensi kemunculan bersama dari kedua kata tersebut dalam keseluruhan *korpus* sedangkan lainnya

local context window methods yang mempelajari hubungan semantik lokal antar kata.

3.3. Transformer

Arsitektur *transformer* yang diusulkan oleh Vaswani [16] dapat mengatasi keterbatasan paralel dari model seperti LSTM (*Long Short Term Memory*) dan RNN (*Recurent Neural Network*) yang bekerja secara sekuensial. RNN dan LSTM membutuhkan memori yang besar karena menyimpan gradient dari proses sebelumnya, selain itu cara kerja sekuensial antar proses membutuhkan waktu training yang lama.



Gambar 3. 1 Arsitektur Model Transformer [16]

Transformer memproses input secara paralel dengan memperkenalkan *attention* sehingga tidak perlu menunggu proses satu selesai sebelum ke proses selanjutnya tapi *transformer*

menghitung secara paralel matriks probabilitas antar *token* hasil dari *softmax*. Selain dapat dijalankan paralel, *transformer* tidak perlu menyimpan gradient dan *hidden state* dari proses sekuensial yang panjang tapi hanya weight yang digunakan dalam menghitung probabilitas yaitu Q (*Query*), K (*Key*), dan V (*Value*)

Encoder terdiri 6 lapisan identik dengan 2 sub-layer dalam setiap layer. Setiap layer selalu diakhiri dengan layer *normalization* dan *residual conection* yang membantu pelatihan melalui *backpropagation* dan menjaga aliran informasi dari layer sebelumnya. Input embedding berupa kumpulan representasi vektor dari setiap *token* kemudian menggunakan *sinusoidal position encoding* untuk menyimpan urutan *token* dalam sebuah sequence.

Layer pertama dengan *multi-head attention* terdiri dari beberapa *head* yang akan fokus terhadap konteks yang berbeda berdasarkan weight pada Q, K, V untuk setiap *head*. Dot product QK^T menunjukkan kemiripan antara vektor Q dengan K dengan memetakan kesamaan arah antar 2 vektor pada ruang vektor. Faktor penskalaan d_k merupakan pembagian jumlah embedding *token* dengan jumlah *head* yang ada. Concatenation dari sejumlah *self-attention* pada persamaan 3.1 kemudian akan dikalikan dengan *weight matrix* dengan dimensi 768 x 768 jika jumlah *embedding token* adalah 768.

$$\text{Attention(Q, K, V)} = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

Q = representasi vektor dari *token* yang ingin difokuskan atau akan dicari *weight attention*

K = representasi vektor dari semua *token* dalam sequence yang dibandingkan dengan Q untuk menghitung skor kesamaan

V = representasi *token* input yang dikalikan dengan *weight attention*

d_k = ukuran dimensi vektor K sebagai faktor penskalaan untuk menjaga stabilitas gradien

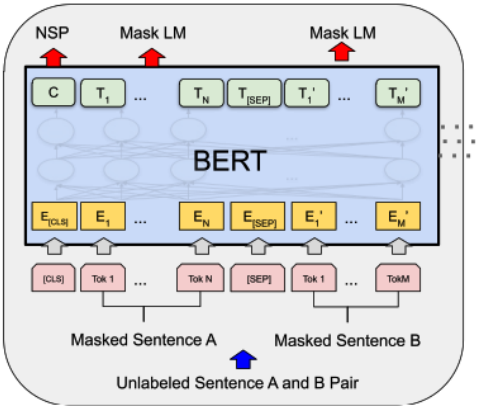
Output dari *self-attention* adalah vektor representasi baru dari setiap *token* yang mencerminkan kontribusi kontekstual dari *token-token* lain dalam kalimat. Vektor ini kemudian diproses oleh layer *feed-forward* yang melakukan transformasi lebih lanjut dan pembaruan bobot untuk tiap *token*.

3.4. BERT

Model BERT [17] merupakan model yang dapat digunakan untuk *word embedding* berbasis arsitektur *transformer*. Pendekatan model *transformer* yang digunakan oleh BERT adalah encoder yang terutama menggunakan *multi-head attention* untuk dapat melakukan sejumlah *self-attention* secara paralel untuk mendapatkan beragam aspek makna sesuai kalimat agar menghindari ambiguitas. Perbedaan utama BERT dengan model non-transformer seperti Word2Vec dan GloVe adalah penggunaan *self-attention* yang dapat mengubah representasi vektor kata dari model *pre-trained* agar menyesuaikan dengan konteks pada kalimat.

Model BERT pertama kali diperkenalkan oleh Google[17] dengan 2 ukuran model yaitu $\text{BERT}_{\text{BASE}}$ dengan 12 *encoder layer*, 12 *attention head*, dan 768 *hidden size* sedangkan $\text{BERT}_{\text{LARGE}}$ menggunakan 24 *encoder layer*, 16 *attention head*, dan 1024 *hidden size*. BERT menggunakan *bidirectional self attention* sehingga dapat memproses kata secara global baik dari kiri maupun kanan. BERT menggunakan *WordPiece embedding* dengan 30.000 *token vocabulary*. *WordPiece embedding* dapat mengatasi keterbatasan *vocabulary* dan kata-kata baru OOV (*Out of*

Vocabulary) dengan memecah *word* menjadi *sub-word* berdasarkan frekuensi kemunculan, setiap *sub-word* lanjutan dari bagian *word* selalu menggunakan "##" di awal bagian *token*. Pada awal setiap sequence selalu diawali dengan special *classification token* [CLS] yang merupakan agregasi dari representasi vektor sejumlah kalimat dalam sequence sedangkan untuk memisahkan antar kalimat dalam sequence menggunakan *token* [SEP] dan memberikan segment *embedding* untuk setiap kalimat.



Gambar 3. 2 Arsitektur BERT [17]

Model *pre-trained* BERT menggunakan *corpus* BooksCorpus dengan 800 juta kata dan English Wikipedia dengan 2.5 miliar kata. Penggunaan *corpus* tingkat dokumen untuk mengambil urutan kata yang panjang dan berkesinambungan (*long contiguous sequences*) yang mempertahanan konteks antar kalimat. Pada saat *pre-training* BERT menggunakan MLM untuk menghasilkan representasi vektor yang dapat memiliki hubungan lokal dalam satu kalimat tapi juga NSP agar dapat menyesuaikan konteks secara global antar kalimat. Kemampuan *bidirectional* dari BERT melalui penggunaan MLM saat *pre-training* dimana *token* diganti secara acak pada 15% dari setiap input sequence, untuk memprediksi *token* tersebut memerlukan BERT untuk

memperhatikan sejumlah *token* baik dari sisi kiri maupun kanan dari [MASK]. Agar BERT dapat lebih *robust* dalam memprediksi *token*, dari 15% *token* acak yang diganti hanya 80% saja *token* [MASK] sebagai sinyal *token* yang diganti sedangkan 10% diganti dengan *token* lain dan 10% lainnya tidak diubah sehingga model BERT belajar memahami konteks tanpa terlalu bergantung dengan sinyal [MASK] saat digunakan untuk *fine-tuning*. NSP digunakan untuk menentukan apakah suatu kalimat merupakan kelanjutan dari kalimat lainnya sehingga model dapat memahami hubungan dan kesinambungan antar kalimat pada sequence yang dipisahkan oleh [SEP] dengan lebih baik.

3.5. Fine Tuning Model BERT

Model BERT meskipun memiliki kemampuan memahami konteks kata secara global sehingga dapat kesulitan untuk generalisasi domain mengingat keterbatasan *corpus* yang digunakan cukup umum. Metode yang dapat dilakukan untuk mengatasi keterbatasan tersebut dengan melatih ulang model *pre-trained* BERT pada domain spesifik jika data yang dimiliki cukup besar serta banyak kata yang belum direpresentasikan (OOV) atau melakukan *fine-tuning* terhadap model *pre-trained* yang sudah ada jika data terbatas dan relatif menghemat waktu dan sumber daya komputasi. Pada penelitian [18] BERT digunakan untuk tugas NLP pada teks dalam mengenali emosi seperti marah, ketakukan, sedih, cinta, gembira, dan terkejut yang didefinisikan oleh [19]. Hal ini menjadi domain yang baru untuk BERT karena model hanya dilatih untuk mengenali konteks dalam suatu sequence sehingga hanya dapat mengklasifikasi polarisasi dari emosi menjadi positif, negatif, dan netral. Pada model [18] BFTE (*BERT embedding concatenated with fine-tuned word embeddings*) melakukan concatenation antara *embedding pre-trained* BERT dengan *fine-tuned word embedding* yang sudah menggunakan emotion vocabulary untuk mengatasi keterbatasan word embedding BERT dimana letak vektor antar kata yang merepresentasikan sebuah

kelas tidak berdekatan dan letak vektor antar kata yang tidak sesuai dengan representasi sebuah kelas justru tidak berdekatan. Membangun *fine-tuned word embedding* dengan *feed forward* neural network berupa sejumlah kata yang memiliki *cosine similarity loss* positif dengan emosi dari sumber ConceptNet dan WordNet. Evaluasi yang didapat model BFTE dengan LSTM atau CNN (*Convolutional Neural Network*) selalu memiliki hasil classification report lebih tinggi dari model lain seperti BERT-CNN dan BERT-LSTM.

Berdasarkan penelitian [20] untuk membuat model untuk analisis sentimen yang dapat adaptif dengan berbagai domain. Model BERT digunakan untuk menghasilkan representasi vektor dari kata yang kemudian ekstraksi fitur lokal oleh TextCNN serta ekstraksi fitur global oleh Bi-LSTM dan BiGRU (*Bidirectional Gated Recurrent Unit*). Output dari model TextCNN, Bi-LSTM, dan BiGRU digunakan sebagai *Query*, *Key*, dan *Value* bagi *self-attention* untuk membuat representasi vektor yang lebih akurat untuk domain dataset. Selain *fine-tuning* melalui penggabungan BERT dengan model lain seperti Bi-LSTM yang dapat memproses secara sekuensial antar vektor, terdapat juga *tuning* model BERT dengan *learning rate* 1e-3 kemudian model BERT-CBLBGA dilatih sebanyak 30 *epoch* dengan *batch size* 16. Dataset yang digunakan dari berbagai domain seperti review film, makanan, dan hotel online agar model *robust* terhadap berbagai subjek. Model menghasilkan akurasi tertinggi sebesar 98,46% dengan output TextCNN sebagai *Query*, BiGRU sebagai *Key*, dan Bi-LSTM sebagai *Value* pada dataset opini media sosial weibo_senti_100k.

Penelitian [21] menggunakan model BERT untuk melakukan analisis sentimen terhadap komentar di Twitter. Model BERT hanya disesuaikan menggunakan *fine-tuning* alih-alih melakukan *pre-training* ulang dari model *transfer learning* BERT. Hal ini karena model BERT dilatih menggunakan *corpus* yang cukup memadai untuk memahami nuansa bahasa yang umum digunakan di media sosial seperti Twitter. Langkah *preprocessing* yang dilakukan dengan transformasi sejumlah emoji, hashtag,

URL, emoticon, mention menjadi bentuk lebih umum misalnya emoji @J_Doellman is back! https://t.co/uBgHGNG34z menjadi user is back! url. Selain itu menambahkan tanda kurung untuk karakterisitik seperti emoji yang ditransformasi. Hal tersebut dapat menghilangkan noise yang disebabkan misalnya mention dan hashtag sehingga struktur teks yang tersusun akan memudahkan model BERT mengekstrak informasi sentimen untuk menentukan representasi vektor yang tepat. Evaluasi yang didapat menunjukkan model yang diusulkan mendapat akurasi tertinggi 0,677 untuk dataset SemEval2017 dan rata-rata F1 score tertinggi 0,7506 untuk dataset SENTIPOLC 2016 dengan spesifikasi model BERT dengan learning rate 3e-5, epoch sebanyak 5, batch size 8, max sequence length 128, dan gradient accumulation step 16.

Penelitian [22] melakukan eksperimen pada bagaimana encoder layer pada BERT berperan dalam berbagai domain NLP downstream. Pengujian dilakukan pada 3 dataset dari benchmark GLUE (*General Language Understanding Evaluation*) yaitu SST-2 untuk analisis sentimen, QNLI untuk tugas NLI (*Natural Language Inference*), dan CoLA untuk Grammatical Acceptability Classification. *Batch size* yang digunakan 8 dan *learning rate* 2e-5. Keseluruhan 12 *encoder layer* pada BERT memiliki arsitektur yang sama tapi berbeda dalam parameter hasil pre-train, yang menyebabkan setiap layer dapat berperan secara fungsional berbeda.

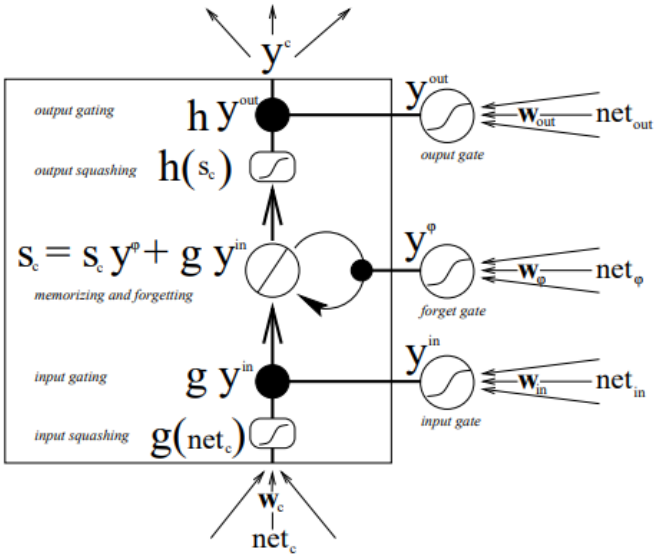
Pengujian dilakukan dengan *reinitialization* dan *probing* sejumlah encoder layer untuk mengetahui kontribusi layer. *Reinitialization encoder layer* secara bertahap dari layer pertama hingga akhir dengan mengganti parameter yang ada dengan weight acak dapat menginformasikan pentingnya peran parameter pada layer tersebut. *Probing* dengan melakukan freeze pada seluruh encoder layer kemudian mengambil representasi hidden state dari masing-masing layer. Classifier linear layer menghasilkan logits dari rata-rata representasi tersebut kemudian menghasilkan probabilitas dengan *softmax* untuk menentukan seberapa informatif representasi dari suatu layer untuk tugas prediksi.

Untuk masing-masing dataset, 4 layer awal sangat penting untuk QNLI dimana performa menurun saat *reinitialization* layer awal, dataset SST-2 mementingkan *middle layers* karena SST-2 membutuhkan pemahaman semantik yang sebagian besar terletak di *middle layers*, dataset CoLa membutuhkan *middle layers* seperti SST-2 tapi persebaran layers lebih merata untuk pemahaman grammar. Selain menentukan layer mana yang penting, urutan *encoder layer* penting karena setelah eksperimen dengan melakukan permutasi urutan *encoder layers* menurunkan performa secara drastis

3.6. Bi-LSTM

Model Bi-LSTM merupakan penggunaan model LSTM [23] dengan arah *forward* dan *backward* dalam memproses input[24]. Masalah *vanishing gradient* atau *exploding gradient* pada RNN yang disebabkan oleh penggunaan fungsi aktivasi *tanh* yang akan memperkecil atau memperbesar nilai gradient terus-menerus sepanjang sequencial state yang panjang dapat diatasi dengan LSTM. LSTM menggunakan *hidden state* (h_t) sebagai *short-term memory* dan juga *cell-state* (c_t) sebagai *long-term memory* sehingga c_t dapat menentukan besarnya informasi state terdahulu yang dapat disimpan. Menentukan bagaimana LSTM dapat mengatur gradient yang ingin digunakan dengan menggunakan *input gate* (i_t), *forget gate* (f_t), dan *output gate* (o_t). i_t menggunakan fungsi aktivasi *sigmoid* (σ) untuk menentukan probabilitas menambahkan gradient dari candidate cell state (g_t) untuk digunakan di c_t. f_t menggunakan σ untuk menentukan probabilitas mempertahankan gradient dari c_{t-1} ke c_t. g_t merupakan gradient informasi dari input state saat ini (x_t) serta informasi *hidden state* sebelumnya h_{t-1} yang akan ditambahkan ke c_t menggunakan aktivasi *tanh*. c_t merupakan *weighted sum* antara probabilitas i_t terhadap g_t dengan probabilitas f_t terhadap c_{t-1} sehingga merupakan penyimpanan memori jangka panjang dari gradient sejumlah c_t terdahulu yang ingin disimpan dan

informasi baru g_t hasil dari cell. o_t menggunakan σ untuk menentukan probabilitas banyaknya informasi dari x_t dan h_{t-1} juga termasuk c_t sehingga dapat tetap memberikan update gradient dengan mempertahankan informasi penting dari state sebelumnya. h_t merupakan besarnya probabilitas o_t akan mengambil gradient dari c_t melalui fungsi aktivasi *tanh* yang akan diteruskan ke timestep berikutnya. Setiap penggunaan aktivasi untuk menghitung gradient maupun membuat keputusan probabilitas menggunakan *weight matrix* W yang disesuaikan dengan variabel dan gate yang digunakan.



Gambar 3. 3 Arsitektur Layer LSTM [23]

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + bi) \tag{3.2}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + bf\right)$$

(3.3)

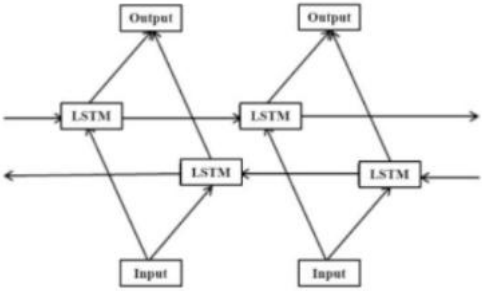
$$g_t = tanh(W_{xc}x_t + W_{hc}h_{t-1} + bc) \tag{3.4}$$

$$c_t = i_t g_t + f_t c_{t-1} \tag{3.5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + bo) \tag{3.6}$$

$$h_t = o_t \cdot tanh(c_t) \tag{3.7}$$

Model LSTM hanya memproses input secara sekuensial dari satu arah saja dimana pada data teks untuk mendapatkan konteks tidak hanya memerlukan timestamp terdahulu tapi juga timestamp yang akan datang. Hasil dari word embedding model seperti BERT merupakan representasi vektor yang memiliki konteks lokal maupun global dari corpus dimana hasil didapat dengan melihat hubungan antar kata dari 2 arah. Penggunaan Bi-LSTM dapat memproses input secara sekuensial seperti LSTM tapi juga memproses input dari 2 arah sehingga Bi-LSTM cocok untuk menangkap hubungan sekuensial dari output BERT yang memiliki makna kontekstual baik lokal maupun global. Hal ini karena BERT tidak secara eksplisit menangkap urutan secara sekuensial tapi menggunakan fungsi *sinusoidal* untuk berbagai posisi *token* dalam sebuah sequence. Memproses output BERT dengan Bi-LSTM diharapkan dapat menangkap pola sekuensial dari hubungan kontekstual antar kata dengan lebih baik dengan tetap mempertahankan makna kontekstual dari vektor hasil output BERT melalui sifat *bidirectional* Bi-LSTM.



Gambar 3. 4 Arsitektur Layer Bi-LSTM [24]

3.7. Metriks Klasifikasi

Metriks yang digunakan untuk evaluasi data testing adalah *accuracy*, *precision*, *recall*, dan *F1-score*. *Accuracy* adalah metriks yang mengukur persentase prediksi yang benar *True Positive* (TP) dan *True Negative* (TN) terhadap keseluruhan hasil prediksi model.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.8}$$

Precision menentukan tingkat kebenaran prediksi positif memang relevan dengan kelas positif pada data aktual sehingga merupakan perbandingan TP dengan TP dan *False Positive* (FP).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.9}$$

Recall mengukur proporsi sampel positif yang berhasil diidentifikasi dengan benar oleh model sehingga merupakan perbandingan TP dengan TP dan *False Negative* (FN).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.10}$$

F1-score merupakan metriks yang mempertimbangkan kedua *precision* dan *recall* ketika data *imbalance*. *F1-score* mendekati 1 jika model memiliki *precision* dan *recall* tinggi sedangkan F1-score mendekati 0 jika salah satu atau kedua *precision* dan *recall* rendah.

$$\text{F1 score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{3.11}$$

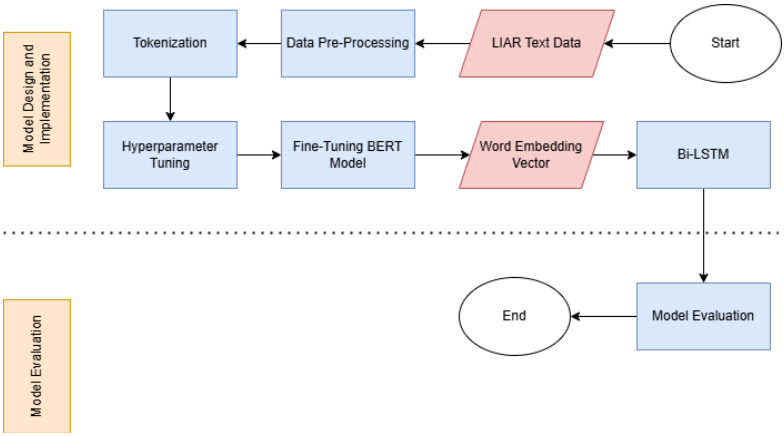
[Halaman ini sengaja dikosongkan]

BAB IV
ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan mengenai sejumlah tahapan yang digunakan dalam penelitian meliputi perancangan sistem dan skenario model serta solusi dan rencana model klasifikasi menggunakan model BERT dan Bi-LSTM

4.1. Desain Sistem

Langkah-langkah yang diperlukan dalam penelitian akan dijabarkan dengan kerangka berpikir dari implementasi model pada Gambar 3.1



Gambar 4. 1 Diagram Alir Sistem

Pada implementasi ini tahap pertama adalah menentukan tahap penelitian. Tahap penelitian yang pertama adalah menentukan ide dari pertemuan dengan dosen pembimbing dan analisis literatur makalah dari jurnal yang berkaitan dengan model klasifikasi teks berita berbasis model *word embedding*. Dari

pertemuan tersebut, solusi yang diperoleh disusun dalam rumusan masalah penelitian atau *research question*.

Tahap selanjutnya adalah tahap implementasi model klasifikasi teks berita berbasis model *word embedding* menggunakan model BERT dan Bi-LSTM. Langkah pertama adalah mendapatkan dataset berisi sejumlah fitur dengan label kebenaran masing-masing yaitu true, mostly-true, half-true, barely-true, false, pants-fire. Dataset yang didapat telah dibagi menjadi bagian training, validation, dan test dengan masing-masing sebanyak 10.240, 1.284, dan 1.267 record. Masing-masing bagian dataset akan melalui proses *pre-processing* berupa label encoding dan mengubah fitur yang relevan menjadi *metadata*. Proses *preprocessing* selanjutnya adalah melakukan *tokenisasi metadata* dan fitur pernyataan tersebut dengan BertTokenizer menjadi sejumlah *token* dari *sub-word* sebuah sequence.

Dalam implementasi model, *token* akan masuk ke model *word embedding* yaitu BERT dengan tipe *bert-base uncased* dimana memiliki 12 *encoder layers*, 768 *hidden size*, dan 12 *attention heads* serta melakukan konversi semua teks ke huruf kecil. Masing-masing *token* akan memiliki 768 *hidden size* yang merepresentasikan nilai vektor sepanjang 768 dimensi ruang vektor. Setiap representasi vektor pada token akan dilakukan *fine-tuning* pada 12 *encoder layers* dengan 12 *attention heads* pada masing-masing *encoder layer*. *Encoder layer* memiliki mekanisme *self-attention* yang akan mengubah representasi vektor *token* dengan melihat kedekatan antar vektor pada 768 dimensi ruang vektor. Peran 12 *attention heads* pada *encoder layer* memungkinkan representasi vektor setiap *token* untuk fokus menangkap 12 hubungan kontekstual berbeda secara paralel yang memperkaya representasi vektor secara global. Hasil dari BERT kemudian akan digunakan sebagai input untuk model Bi-LSTM dengan 2 layer yang diakhiri dengan classifier layer yang menghasilkan jumlah output neuron sesuai dengan jumlah kelas pada skenario klasifikasi (2 atau 6 kelas). Selama proses training dan validasi, loss dihitung menggunakan Cross Entropy Loss yang

membandingkan hasil logit dari output classifier layer (sebelum *softmax*) dengan label yang benar dalam klasifikasi multi kelas.

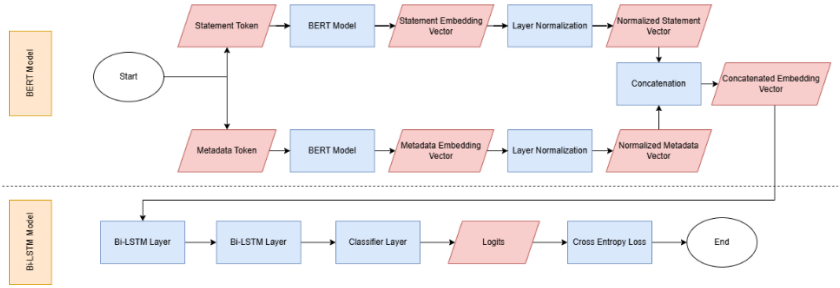
Langkah terakhir adalah evaluasi performa model dengan classification reports yang menampilkan accuracy, precision, recall, dan F1-score serta menggunakan confusion matrix untuk jumlah kesesuaian kelas sebenarnya dengan kelas prediksi.

4.2. Desain Model

Pada bagian ini akan menggambarkan bagaimana desain model dari eksperimen model arsitektur BERT dengan Bi-LSTM. Setiap hasil *tokenisasi* dari training, validation, dan test akan memiliki *batch size* yang sama yaitu 16. Selama proses *fine-tuning* akan dilakukan beberapa skenario dengan melakukan *freeze* pada sejumlah *encoder layer* awal sesuai skenario pengujian. Selain *fine-tuning encoder layer* juga *hyperparameter tuning* untuk *learning rate* dari model BERT sesuai dengan skenario pengujian. Model pre-trained BERT bert-base uncased akan menerima input hasil dari *tokenisasi* sequence tersebut yang kemudian digunakan untuk menghasilkan output last_hidden_state. Output last_hidden_state tersebut merupakan representasi vektor dari setiap *token* pada setiap sequence sehingga jika menggunakan *bert-base uncased* maka shape dari last_hidden_state adalah [jumlah_sequence, panjang_sequence, 768]. last_hidden_state sering digunakan daripada keseluruhan output bert setiap *hidden state* atau *encoder layer* karena vektor pada last_hidden_state sudah melalui update *weight* hingga *encoder layer* ke-12 yang dianggap sudah memiliki akumulasi informasi dari *encoder layer* sebelumnya.

Output dari BERT berupa vektor yang ada pada last_hidden_state akan dinormalisasi menggunakan LayerNormalization untuk menjaga stabilitas pelatihan dan meningkatkan kemampuan generalisasi model Bi-LSTM. Kedua last_hidden_state yaitu dari *metadata* dan pernyataan digabungkan menggunakan torch.cat menghasilkan shape [jumlah_sequence, panjang_sequence, 1536]. Vektor hasil concat last_hidden_state

tersebut kemudian menjadi input untuk layer Bi-LSTM dengan input size dan *hidden state* yang sesuai dengan ukuran dimensi *embedding* dari vektor last_hidden_state yaitu 768 sehingga tidak terjadi *bottleneck* atau menambah noise. Bi-LSTM disini menggunakan 2 layer saja agar tidak menambah kompleksitas dan menghindari potensi *overfitting*. Output dari Bi-LSTM memiliki 2*dimensi *embedding* karena sifat *bidirectional*. Dari output tersebut kemudian menghasilkan *logits* seukuran kelas klasifikasi teks berita melalui *classifier layer*. *Logits* tersebut akan diubah ke probabilitas multi kelas menggunakan *softmax* pada *Cross Entropy Loss* untuk pembaruan *weight* selama *backpropagation*. *Epoch* digunakan selama training dan validasi sesuai dengan skenario pengujian.



Gambar 4. 2 Desain Model Bi-LSTM dan BERT

4.3. Dataset

Dataset yang digunakan dalam penelitian ini adalah LIAR Fake News Dataset yang merupakan dataset pendeteksian berita palsu dengan 6 kelas. Terdapat total 12.791 sample dalam dataset untuk keseluruhan kelas. Distribusi sample kelas dapat dilihat pada Tabel 4.1

Tabel 4. 1 Distribusi Sample Untuk 6 Kelas

	Sample Training	Sample Validasi	Sample Testing
True	1676	169	208
Mostly-true	1962	251	241
Half-true	2114	248	265
Barely-true	1654	237	212
False	1995	263	249
Pants-fire	839	116	92

Penelitian ini menggunakan 6 kelas seperti keadaan awal dataset dan juga mengelompokkan kelas “true” dan “mostly-true” sebagai “true” sedangkan kelas lainnya yaitu “half-true”, “barely-true”, “false”, dan “pants-fire” sebagai “false” menghasilkan hanya 2 kelas. Distribusi untuk hanya 2 kelas dapat dilihat pada Tabel 4.2

Tabel 4. 2 Distribusi Sample Untuk 2 Kelas

	Sample Training	Sample Validasi	Sample Testing
True	3638	420	449
False	6602	864	818

4.4. Pre-Processing Data

Pada tahap pre-processing data dilakukan proses *fetch* dataset yang telah dibagi menjadi train, valid, dan test. Langkah selanjutnya dengan melakukan *drop* record yang memiliki null values. Selain itu, mengingat keterbatasan sumber daya komputasi dan waktu maka hanya menggunakan 40% data dari training untuk keseluruhan distribusi kelas

Proses pre-processing dilakukan agar model word embedding dapat memproses input berupa teks menjadi nilai numerik yang informatif tentang hubungan kontekstual dari fitur pada dataset. Langkah-langkah *preprocessing* termasuk

melakukan *label encoding* untuk mengubah label kelas menjadi format numerik yang dapat diproses oleh model, konstruksi *metadata* dari fitur informatif, dan *tokenisasi* untuk mengubah *metadata* yang diperoleh menjadi sejumlah *token* yang memiliki representasi vektor informatif tentang hubungan kontekstual antar *sub-word*

4.4.1. Label Encoding

Label encoding merupakan proses transformasi label kelas yang semula berupa *string* menjadi format numerik agar model dapat dievaluasi menggunakan *Cross Entropy Loss* berdasarkan index label kelas. *Label encoding* untuk klasifikasi biner menggunakan *list comprehension* dimana label kelas "true" dan "mostly-true" diberi lable "1" sedangkan class lainnya yaitu "half-true", "barely-true", "false", dan "pants-fire" diberi label "0".

Label encoding untuk 6 kelas dilakukan menggunakan np.select yang dapat melakukan mapping nilai berdasarkan sejumlah kondisi dan nilai pada parameter. Parameter condlist berupa kondisi jika fitur label class pada dataset sesuai dengan *string* dari label kelas sedangkan parameter choicelist berisi mapping nilai numerik label kelas berdasarkan kecocokan dengan nama *string* label kelas. Lable dengan *string* "true" diberi lable numerik "0", "mostly-true" diberi lable "1", "half-true" diberi lable "2", "false" diberi lable "3", "barely-true" diberi lable "4", "pants-fire" diberi lable "5"

4.4.2. Konstruksi Metadata

Tidak keseluruhan fitur pada dataset digunakan karena relevansi dan perlunya *tuning* untuk dengan tepat memanfaatkan

fitur tersebut sehingga tidak berpotensi memperburuk hasil model. Hanya 7 dari 14 fitur yang digunakan untuk konstruksi *metadata* termasuk fitur berisi pernyataan dari teks berita yang penting untuk hubungan kontekstual. Fitur lain tidak digunakan seperti fitur berisi ID dari statement dan juga label kelas. Selain itu fitur numerik mengenai jumlah suatu pernyataan diklasifikasikan menjadi "barely-true", "false", "half-true", "mostly-true", dan "pants-fire" tidak digunakan karena ketidakkonsistenan yang besar dari jumlah antar pernyataan dan juga meskipun diubah ke persentase akan sulit untuk menentukan weight dari persentase setiap jumlah.

Fitur yang digunakan adalah subjek atau tema, pembicara, profesi pembicara, negara bagian dari pembicara, afiliasi partai politik dari pembicara, dan konteks dari teks berita. Dari fitur yang digunakan tersebut digabungkan dengan fungsi join dimana jika nilai fitur sama dengan 0 maka diganti dengan "None". Hasilnya adalah metadata dengan setiap record berisi setiap fitur tersebut yang dijeda dengan " "

4.4.3. Tokenisasi

Tokenisasi adalah proses mengubah data teks yang berupa sequence menjadi sejumlah *sub-word* yang direpresentasikan sebagai *token* yang memiliki ID. Hal ini bertujuan agar model dapat membuat representasi vektor berdasarkan ID *token* yang sudah ada pada *vocabulary pre-trained* BERT tepatnya *bert-base uncased*. Proses tokenisasi menggunakan BertTokenizer dari library transformers.

Metadata dan fitur pernyataan sebelumnya akan ditransformasi menghasilkan output yang berisi informasi input_ids merupakan ID dari *token* termasuk *padding token*, [CLS],

dan [SEP]. Output juga berisi attention_mask merupakan informasi apakah suatu *token* termasuk *padding token* dengan nilai 0 dan selain *padding token* bernilai 1. *Padding token* ditambahkan jika *token* tidak mencapai jumlah max_length yaitu 512. Jika melebihi max_length maka token yang melebihi max_length akan dipotong. Hasilnya adalah output tensor dengan shape [jumlah_sequence, 1, 512] sehingga perlu di-reshape menggunakan torch.squeeze(1) untuk membuat shape-nya menjadi [jumlah_sequence, 512] sebelum digunakan untuk input model BERT.

4.5. Proses Training dan Validasi

Proses training dan validasi dilakukan untuk mendapatkan hasil dari kinerja model klasifikasi berita palsu menggunakan model BERT dan Bi-LSTM. Dengan inisiaslisasi *hyperparameter* dan kemudian training dan testing berulang kali berdasarkan skenario pengujian.

4.5.1. Insialisasi Hyperparameter

Inisialisasi *hyperparameter* bertujuan untuk menentukan parameter yang sesuai agar performa model optimal. Model dilatih sebanyak 4 atau 6 epoch. Dengan jumlah iterasi tersebut diharapkan model dapat menyesuaikan weight dari masing-masing bagian parameter untuk meningkatkan akurasi prediksinya. Meningkatkan akurasi dengan mengetahui banyak loss dari setiap iterasi menggunakan *Cross Entropy Loss* karena prediksi multi label kelas. Dari loss yang didapat kemudian update *weight* melalui *backpropagation* dilakukan oleh *optimizer AdamW* varian dari *Adaptive Moment Estimation* (Adam) yang memisahkan *weight decay* dari update gradient sehingga selain mempercepat

konvergensi dan stabilitas pelatihan melalui *momentum adaptif learning rate*. AdamW menggunakan *weight decay* langsung untuk melakukan update parameter sehingga tidak berefek ke gradient. Hal ini membuat optimisasi dan generalisasi lebih baik dari Adam. *Learning rate* untuk model Bi-LSTM dan *LayerNormalization* diatur pada 1e-3. Nilai *dropout* dari hasil *LayerNormalization* diatur pada 0,1 berdasarkan BertConfig.hidden_dropout_prob untuk mencegah overfitting. Selama proses training dan validasi, dropout akan membuat model tidak bergantung ke neuron dengan menonaktifkan sejumlah neuron secara acak

4.5.2. Skenario Pengujian

Selama pengujian akan dilakukan empat skenario dengan keterangan sebagai berikut:

a. Jumlah klasifikasi label menggunakan 2 dan 6 label kelas
b. Learning rate model BERT diatur pada 2e-5 dan 3e-5
c. Jumlah encoder layer awal yang di-freeze yaitu hingga layer ke-2, 4, dan 6
d. Jumlah epoch selama training dan validasi dari 3 hingga 6

4.6. Evaluasi Model

Setelah menguji berbagai skenario melalui proses panjang training dan validasi kemudian proses testing model untuk menghasilkan evaluasi. Evaluasi dilakukan dengan menggunakan classification reports yang berisi accuracy, precision, recall, dan F1-score. Selain itu juga confusion matrix yang menampilkan jumlah lable prediksi dan lable yang sebenarnya

[Halaman ini sengaja dikosongkan]

BAB V
IMPLEMENTASI

Bab ini membahas tentang implementasi dari model yang dibagi menjadi preprocessing data, proses training dan testing, serta evaluasi model.

5.1 Preprocessing

Berikut implementasi pseudocode pada proses preprocessing

5.1.1 Ekstraksi Metadata

Berikut implementasi pseudocode pada proses ekstraksi *metadata*

Algorithm 1: Extract Metadata

```
Function preprocessing(dataset, num_labels):
    if num_labels == 2 then
        foreach idx, record in enumerate(dataset[1]) do
            if record == "true" or record == "mostly-true" then
                dataset['labels'][idx] = 1
            else
                dataset['labels'][idx] = 0
    else if num_labels == 6 then
        cond = [
            (dataset[1]=="true"),
            (dataset[1]=="mostly-true"),
            (dataset[1]=="half-true"),
            (dataset[1]=="false"),
            (dataset[1]=="barely-true"),
            (dataset[1]=="pants-fire")
        choice = [0,1,2,3,4,5]
        dataset['labels']= select_labels(cond, choice, default=0)
        dataset = drop_columns(dataset, [0,1,8,9,10,11,12])
        dataset = reset_index(dataset)
        Initialize metadata as empty list
        for i = 0 to length(dataset) do
            Initialize temp as empty string
            for each col in [3,4,5,6,7,13] do
                if dataset.loc[i, data] != 0 then
                    temp += " " + dataset.loc[i, col]
                else
                    temp += " None"
            Add temp to metadata
        dataset[14] = metadata
        dataset['meta'] = str(dataset[14])
        dataset['sentence'] = dataset[2]
        dataset = drop_columns(dataset, [2,3,4,5,6,7,13,14])
        return dataset
```

Pseudocode 5. 1 Ekstraksi Metadata

Pada Pseudocode 5.1 melakukan ekstraksi sejumlah fitur untuk menjadi *metadata*. Langkah diawali dengan membuat label untuk masing-masing jumlah kelas. Fitur-fitur yang digunakan meliputi topik, speaker, profesi speaker, negara, afiliasi partai untuk *metadata* dan juga pernyataan dari speaker.

5.1.2 Tokenisasi

Berikut implementasi pseudocode proses *tokenisasi*

Algorithm 2: Tokenization

```
Class text_dataset (Dataset):
    Procedure __init__:
        self.statement = dataset['sentence']
        self.meta = dataset['meta']
        self.labels = dataset['labels']
        self.tokenizer = BertTokenizer("bert-base-uncased")
    Function __getitem__:
        state_encoding = tokenizer_encoding(
            str(self.statement[idx]),
            max_length = 512,
            padding = "max_length",
            return_attention_mask = True,
            truncation = True,
            return_tensors = 'pt'
        )
        meta_encoding = tokenizer_encoding(
            str(self.meta[idx]),
            max_length = 512,
            padding = "max_length",
            return_attention_mask = True,
            truncation = True,
            return_tensors = 'pt'
        )
        return meta_encoding['input_ids'],
          meta_encoding['attention_mask'], state_encoding['input_ids'],
          state_encoding['attention_mask'], self.labels[idx]
    Function __len__(self):
        return length(self.statement)
```

Pseudocode 5. 2 Proses Tokenisasi

Pseudocode 5.2 menggunakan BertTokenizer dari Hugging Face untuk melakukan tokenisasi pada input pernyataan dan *metadata*. Keduanya menggunakan parameter yang sama

dimana setiap record akan memiliki panjang 512 token dimana diisi nilai padding hingga 512 jika kurang dari jumlah tersebut atau dipotong jika melebihi. Hasil dari tokenisasi adalah input_ids dan attention_mask. Bagian input_ids merupakan id dari *sub-word* yang ada pada *vocabulary* model BERT. Bagian attention_mask berisi angka 0 jika id pada input_ids merupakan padding dan 1 jika sebaliknya.

5.2 Model

Berikut implementasi dari model yang digunakan dalam eksperimen yaitu BERT dan Bi-LSTM:

5.2.1 Model Bi-LSTM dan Classifier

Berikut implementasi pseudocode untuk model Bi-LSTM dan Classifier

Algorithm 3: Bi-LSTM Model

```
Class BiLSTM:
    Procedure __init__(self, hidden_dim, embedding_dim,
    label_size):
        Initialize BiLSTM_layer with:
            input_size=embedding_dim
            hidden_size=hidden_dim
            num_layers=2
            bidirectional=True
        Initialize LSTM_layer with:
            in_features=hidden_dim*2
            out_features=label_size
        self.hidden_dim = hidden_dim
        self.lstm = BiLSTM_layer
        self.classifier = LSTM_layer
    Function forward(self, bert_embedding):
        lstm_out, _ = self.lstm(bert_embedding)
        logits = self.classifier(lstm_out)
        return logits
```

Pseudocode 5. 3 Model Bi-LSTM dan Classifier

Model Bi-LSTM pada Pseudocode 5.3 terdiri dari 2 layer Bi-LSTM yang menerima input dan memiliki hidden layer seukuran dengan 2 kali output model BERT yaitu 768, masing-masing dari metadata dan pernyataan sehingga menjadi 1536.

Selesai dari Bi-LSTM kemudian masuk ke classifier menggunakan LSTM yang menghasilkan output sebanyak jumlah label

5.2.2 Model BERT dan Bi-LSTM

Berikut pseudocode untuk model BERT dan Bi-LSTM

Algorithm 4: BERT and Bi-LSTM Model

```
Class Bert_BiLSTM:
    Procedure __init__(self, num_layer, num_labels):
        Initialize Layer_Normalization with:
            normalized_shape=self.bert.config.hidden_size
            eps=1e-12
        self.num_labels = num_labels
        self.num_layer = num_layer
        self.bert = BertModel('bert-base-uncased')
        self.layerNorm = Layer_Normalization
        self.dropout = Dropout(p=self.bert.config.hidden_dropout_prob)
        self.classifier = BiLSTM(self.bert.config.hidden_size*2,
         self.bert.config.hidden_size*2, self.num_labels)
        init_weight(self.classifier.classifier.weight
        foreach name, param in self.classifier.lstm.named_parameters()
         do
            if 'weight' in name then
                init_weight(param)
        self.set_encoder()
    Function forward(self, metaID, metaMask, stateID,
     stateMask):
        metadata = self.bert(reduce_dimension(metaID, 1),
         reduce_dimension(metaMask, 1)
        metaNorm =
         self.dropout(self.layerNorm(metadata.last_hidden_state[:, 0, :]))
        sentence = self.bert(reduce_dimension(stateID, 1),
         reduce_dimension(stateMask, 1)
        sentenceNorm =
         self.dropout(self.layerNorm(sentence.last_hidden_state[:, 0, :]))
        output = concatenation((metaNorm,sentenceNorm), -1)
        logits = self.classifier(output)
        return logits
    Procedure set_encoder(self):
        foreach param in self.bert.parameters() do
            param.requires_grad = True
        for i = 0 to self.num_layer do
            foreach param in self.bert.encoder.layer[i].parameters() do
                param.requires_grad = False
```

Pseudocode 5. 4 Model BERT-BiLSTM

Dari Pseudocode 5.4 menjelaskan bagaimana model Bi-LSTM dan BERT digunakan hingga menghasilkan logits. Token CLS hasil output BERT yang menerima dari input token pernyataan dan *metadata*, kemudian akan digabungkan sehingga

memiliki dimensi terakhir 1536. *Token* gabungan tersebut kemudian dinormaliasi pada layer *BatchNormalization*. Hasil output terakhir diklasifikasikan menggunakan model Bi-LSTM menghasilkan *logits*

5.3 Proses Training dan Testing

Berikut implementasi dari proses training dan testing:

5.3.1 Training

Algorithm 5: Model Training

```
Function train_model(num_layer, bert_lr, num_labels):
    model = Bert_BiLSTM(num_layer, num_labels)
    num_epoch = [3,4,5,6]
    Initialize AdamW with:
        {params: model.bert.parameters(), lr: bert_lr}
        {params: model.layerNorm.parameters(), lr: 0.001}
        {params: model.classifier.parameters(), lr: 0.001}
    optimizer = AdamW
    Initialize criterion as CrossEntropyLoss()
    best_model = None
    best_acc = 0
    time_start = time.time()
    for epoch = 0 to max(num_epoch) do
        val_loss, val_acc, val_total ← 0
        foreach phase in [train, val] do
            if phase == train then
                model.train()
            else
                model.eval()
            foreach batch in dataloaders_dict[phase][str(num_labels)] do
                meta, state
                labels = batch
                metaID = meta[input_ids]
                metaMask = meta[attention_mask]
                stateID = state[input_ids]
                stateMask = state[attention_mask]
                optimizer.zero_grad()
                outputs = model(metaID, metaMask, stateID, stateMask)
                loss = criterion(outputs, labels)
                if phase == train then
                    loss.backward()
                    optimizer.step()
                else if phase == val then
                    val_loss += loss.item()
                    predicted = find_max_index(outputs, 1)
                    val_total += labels.size(0)
                    val_correct += sum(predicted == labels)
            val_acc = val_correct / val_total
            total_val_loss = val_loss / len(dataloader[val])
        Print("Epoch {epoch+1}/{max(num_epoch)}
        ||Validation Loss : {total_val_loss : .4f}||Validation Acc :
        {val_acc : .4f}")
        if epoch+1 in num_epoch then
            elapse = time.time() - time_start
            Print "Training complete{in {elapse // 60:.0f}m {elapse %
              60:.0f}"
            Print "Frozen BERT Layers: 0 to {num_layer-1}"
            Print "Epoch: {epoch+1}"
            Print "Learning Rate: {bert_lr}"
            acc = evaluate(model, num_labels=num_labels)
            if acc > best_acc then
                best_acc = acc
                best_model = copy_model(model)
    return best_model, best_acc
```

Pseudocode 5. 5 Proses Training

Berdasarkan Pseudocode 5.5 proses training diawali dengan inisialisasi model BERT dan Bi-LSTM, *optimizer AdamW*, dan *criterion* menggunakan *CrossEntropyLoss*. Melakukan training dengan *epoch* maksimal 6 dimana pada setiap *epoch* memproses *batch* berisi hasil *tokenisasi* dari pernyataan dan *metadata*. Pada setiap *batch*, model akan menerima input_ids dan attention mask sebagai input kemudian menghasilkan *logits*. Indeks *logits* terbesar dianggap sebagai label yang diprediksi. Selama validasi akan menghasilkan val_loss dan juga val_acc untuk monitoring kualitas prediksi.

5.3.2 Testing

Berikut implementasi pseudocode dari proses testing:

Algorithm 6: Evaluate Model

```
1  Function evaluate(model, num_labels):
2      model.eval()
3      Initialize predsList as empty list
4      Initialize labelsList as empty list
5      with no_gradient:
6          foreach batch in dataloaders_dict['test'][str(num_labels)] do
7              meta, state
8              labels = batch
9              TmetaID = meta[input_ids]
10             TmetaMask = meta[attention_mask]
11             TstateID = state[input_ids]
12             TstateMask = state[attention_mask]
13             outputs = model(TmetaID, TmetaMask, TstateID,
                 TstateMask)
14             preds = find_max_index(outputs, dim=1)
15             Add preds to predsList
16             Add Tlabels to labelsList
17     show_classification_report(labelsList, predsList)
18     show_confusion_matrix(labelsList, predsList)
19     acc = accuracy_score(labelsList, predsList)
20     precision = precision_score(labelsList, predsList)
21     recall = recall_score(labelsList, predsList)
22     f1 = f1_score(labelsList, predsList)
23     Print "Accuracy : {acc:.4f}"
24     Print "Precision: {precision:.4f}"
25     Print "Recall : {recall:.4f}"
26     Print "F1-Score : {f1:.4f}"
27     return acc
```

Pseudocode 5. 6 Proses Testing

Dari proses testing pada Pseudocode 5.6 akan menampilkan classification report berisi akurasi, presisi, recall, dan F1-score serta confusion matrix dari setiap batch.

5.3.3 Hasil

Berikut implementasi pseudocode dari hasil testing berbagai skenario:

Algorithm 7: Model Result

```
Procedure Result(num_labels):
    best_model = None
    best_acc = 0
    num_layers = [2,4,6]
    bert_lr = [2e-5, 3e-5]
    foreach layer in num_layers do
        foreach lr in bert_lr do
            model, acc = train_model(layer, lr, num_labels)
            if acc > best_acc then
                best_acc = acc
                best_model = copy_model(model)
    model_path = "best_model_num_labels.pth"
    save_model(best_model, model_path)
```

Pseudocode 5. 7 Evaluasi Skenario

[Halaman ini sengaja dikosongkan]

BAB VI
HASIL DAN PEMBAHASAN

6.1. Hasil Evaluasi

Berikut adalah hasil evaluasi dari masing-masing 6 skenario untuk klasifikasi 2 kelas dan 6 kelas dari proses training dan testing

6.1.1. Klasifikasi 2 Kelas

6.1.1.1. Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 1

Performa model pada skenario tampak baik dan dengan kenaikan signifikan pada epoch 5 dengan *accuracy* terbaik 66,51%

Tabel 6. 1 Hasil Evaluasi Model dengan Label 2 Kelas pada skenario Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 1

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,6159	0,6159	0,6086	0,6114
4	0,6066	0,6066	0,6328	0,6127
5	0,6651	0,6651	0,6594	0,6614
6	0,6206	0,6206	0,6331	0,6249

6.1.1.2. Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 3

Performa model pada skenario tampak baik dan konsisten dengan akurasi terbaik 66,98% pada epoch terakhir

Tabel 6. 2 Hasil Evaluasi Model dengan Label 2 Kelas pada skenario Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 3

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,6557	0,6557	0,6448	0,6467
4	0,6534	0,6534	0,6438	0,6461
5	0,6674	0,6674	0,6701	0,6686
6	0,6698	0,6698	0,6663	0,6677

6.1.1.3. Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 5

Performa model pada skenario tampak baik dan *accuracy* terbaik 65,57% pada epoch 4

Tabel 6. 3 Hasil Evaluasi Model dengan Label 2 Kelas pada skenario Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 5

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,6300	0,6300	0,6328	0,6313
4	0,6557	0,6557	0,6618	0,6582
5	0,6230	0,6230	0,6431	0,6285
6	0,6042	0,6042	0,5989	0,6012

6.1.1.4. Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 1

Performa model pada skenario tampak baik dan konsisten dari segi *accuracy* dengan tertinggi 62,53% pada epoch 3,4,6 tapi memiliki *precision* yang cukup rendah dengan 39,10%.

Tabel 6. 4 Hasil Evaluasi Model dengan Label 2 Kelas pada skenario Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 1

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,6253	0,6253	0,3910	0,4811
4	0,6253	0,6253	0,3910	0,4811
5	0,6230	0,6230	0,5714	0,5138
6	0,6253	0,6253	0,3910	0,4811

6.1.1.5. Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 3

Performa model pada skenario tampak baik dengan *accuracy* tertinggi sebesar 64,87% pada epoch awal 3 tapi terjadi penurunan cukup signifikan pada epoch 4. Pada skenario ini konsisten menghasilkan *precision* yang tinggi.

Tabel 6. 5 Hasil Evaluasi Model dengan Label 2 Kelas pada skenario Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 3

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,6487	0,6487	0,6534	0,6507
4	0,5855	0,5855	0,6662	0,5850
5	0,6183	0,6183	0,6411	0,6241
6	0,6300	0,6300	0,6273	0,6285

6.1.1.6. Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 5

Performa model pada skenario tampak baik dan cukup konsisten dengan *accuracy* tertinggi sebesar 64,64% pada epoch 5 dan 6

Tabel 6. 6 Hasil Evaluasi Model dengan Label 2 Kelas pada skenario Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 5

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,6300	0,6300	0,6358	0,6324
4	0,6206	0,6206	0,6235	0,6219
5	0,6464	0,6464	0,6309	0,6312
6	0,6464	0,6464	0,6257	0,6126

Dari keseluruhan skenario untuk hanya label 2 kelas, hasil paling baik diperoleh pada skenario *learning rate* 2e-5 dan *freeze encoder layer* 0 hingga 3 dimana hasil *accuracy* dan *precision* tertinggi masing-masing 66,98% dan 66,63% pada epoch 6 bahkan *accuracy* terendah tetap tinggi dengan 65,34%. Pada skenario tersebut konsisten menghasilkan *precision* dan *recall* yang tinggi sehingga *F1-score* selalu tinggi dengan tertinggi 66,86% dan terendah 64,61%.

6.1.2. Klasifikasi 6 Class

6.1.2.1. Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 1

Performa model pada skenario cukup rendah dan terjadi peningkatan cukup signifikan dengan *accuracy* tertinggi 27, 40%

pada epoch 4. Pada skenario ini *precision* selalu lebih rendah dan terdapat perbedaan cukup signifikan dengan *recall*

Tabel 6. 7 Hasil Evaluasi Model dengan Label 6 Kelas pada skenario Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 1

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,2342	0,2342	0,1456	0,1607
4	0,2740	0,2740	0,1680	0,2012
5	0,2412	0,2412	0,1911	0,2015
6	0,2365	0,2365	0,1697	0,1896

6.1.2.2. Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 3

Performa model pada skenario rendah dengan akurasi tertinggi 21,31% pada epoch 4 dan 6. Presisi pada skenario ini sangat rendah dengan tertinggi 10,50% berakibat pada F1-score yang juga rendah

Tabel 6. 8 Hasil Evaluasi Model dengan Label 6 Kelas pada skenario Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 3

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,2014	0,2014	0,1050	0,0947
4	0,2131	0,2131	0,0454	0,0749
5	0,1897	0,1897	0,0607	0,0741
6	0,2131	0,2131	0,0900	0,0952

6.1.2.3. Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 5

Performa model pada skenario cukup rendah dan konsisten dengan terjadi peningkatan cukup signifikan pada epoch 4. *Accuracy* tertinggi 27,40% terjadi pada epoch 6.

Tabel 6. 9 Hasil Evaluasi Model dengan Label 6 Kelas pada skenario Learning Rate 2e-5 dan Freeze Encoder Layer 0 hingga 5

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,2436	0,2436	0,1146	0,1405
4	0,2717	0,2717	0,3218	0,1577
5	0,2693	0,2693	0,1736	0,2088
6	0,2740	0,2740	0,2128	0,2271

6.1.2.4. Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 1

Performa model pada skenario cukup rendah dan konsisten dengan terjadi peningkatan signifikan pada epoch 6. *Accuracy* tertinggi 26,70% terjadi pada epoch 6.

Tabel 6. 10 Hasil Evaluasi Model dengan Label 6 Kelas pada skenario Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 1

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,2389	0,2389	0,0990	0,1400
4	0,2295	0,2295	0,1356	0,1584
5	0,2272	0,2272	0,3354	0,1577
6	0,2670	0,2670	0,2691	0,2004

6.1.2.5. Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 3

Performa model pada skenario cukup rendah dan konsisten dimana terjadi peningkatan cukup signifikan pada epoch 5 dengan *accuracy* tertinggi 27,87%

Tabel 6. 11 Hasil Evaluasi Model dengan Label 6 Kelas pada skenario Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 3

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,2600	0,2600	0,1076	0,1522
4	0,2482	0,2482	0,1995	0,1818
5	0,2787	0,2787	0,1660	0,2060
6	0,2670	0,2670	0,1653	0,2004

6.1.2.6. Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 5

Performa model pada skenario rendah dengan *accuracy* tertinggi 22,95% pada epoch 5. *Precision* pada skenario ini sangat rendah dengan tertinggi 9,90% berakibat pada *F1-score* yang juga rendah

Tabel 6. 12 Hasil Evaluasi Model dengan Label 6 Kelas pada skenario Learning Rate 3e-5 dan Freeze Encoder Layer 0 hingga 5

Epoch	Accuracy	Recall	Precision	F1-Score
3	0,2131	0,2131	0,0454	0,0749
4	0,1733	0,1733	0,0300	0,0512
5	0,2295	0,2295	0,0990	0,1171
6	0,2131	0,2131	0,0454	0,0749

Dari keseluruhan skenario, performa terbaik terdapat pada skenario learning rate 3e-5 dan freeze encoder layer 0 hingga 3 dimana hasil akurasi dan presisi tertinggi masing-masing 27,87% dan 16,60% pada epoch 5 dan bahkan akurasi terendah 24,82% yang cenderung lebih tinggi daripada mayoritas skenario lainnya menggunakan label 6 kelas.

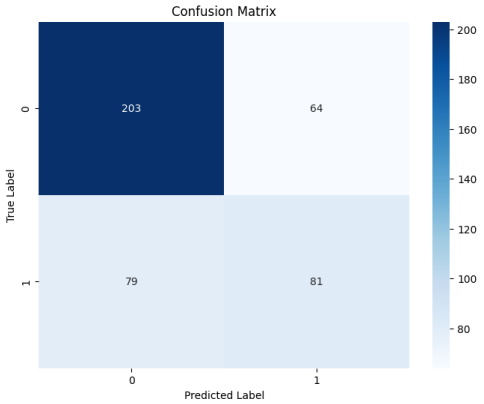
6.2. Pembahasan

Pada bagian ini akan membahas perbandingan sejumlah skenario berdasarkan jumlah *encoder layer* BERT yang *freeze* dan juga *learning rate* dari model BERT

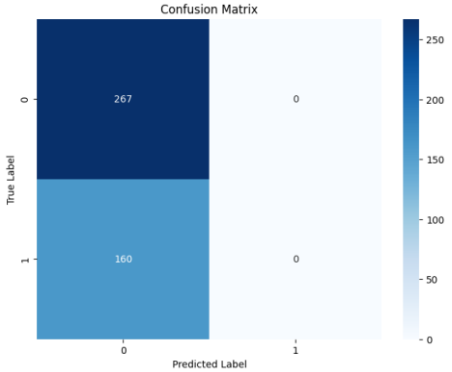
Tabel 6. 13 Hasil Evaluasi Learning Rate untuk Label 2 Kelas dari Epoch Terbaik pada Skenario Freeze Layer 0 hingga 1

Learning Rate	Accuracy	Recall	Precision	F1-Score
2e-5	0,6651	0,6651	0,6594	0,6614
3e-5	0,6253	0,6253	0,391	0,4811

Tabel 6.13 merupakan daftar performa model saat *freeze encoder layer* 0 hingga 1 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*.



Gambar 6. 1 Confusion Matrix Skenario Label 2 Kelas dengan Learning Rate 2e-5 dan Freeze Layer 0 hingga 1



Gambar 6. 2 Confusion Matrix Skenario Label 2 Kelas dengan Learning Rate 3e-5 dan Freeze Layer 0 hingga 1

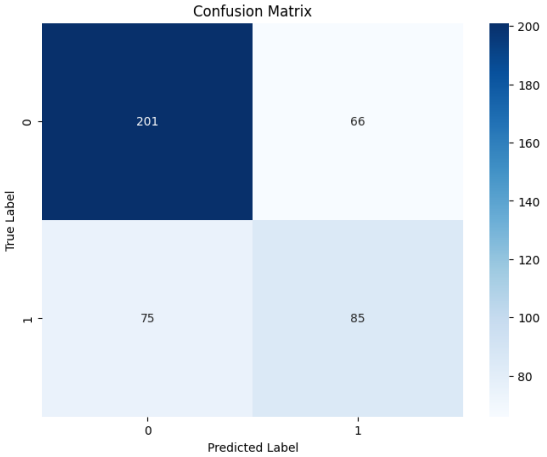
Performa terbaik ditunjukkan saat learning rate 2e-5 dimana *accuracy* dan *recall* cukup tinggi sebesar 66,51% yang berarti

model cukup baik mengenali label sebenarnya sedangkan untuk *learning rate* 3e-5 accuracy dan *recall* tidak berbeda signifikan yaitu 62,53%. Dari metriks lainnya yaitu *precision* dan *F1-score* yang terbaik menggunakan *learning rate* 2e-5 dimana *precision* hanya berbeda sedikit dari *recall* yaitu 65,94% yang berarti model hanya sedikit melakukan salah prediksi yang berakibat pada *F1-score* yang tinggi yaitu 66,14% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada *learning rate* 3e-5 *precision* berbeda cukup jauh dengan *recall* yaitu 39,1% sehingga *F1-score* menjadi rendah hanya 48,11% yang berarti tidak seimbang hanya baik pada satu aspek yaitu menangkap label sebenarnya tapi kurang dalam memprediksi kelas. Pada *learning rate* 3e-5 terlihat dari *confusion matrix* model mengalami *underfitting* dengan hanya memberikan prediksi pada salah satu kelas.

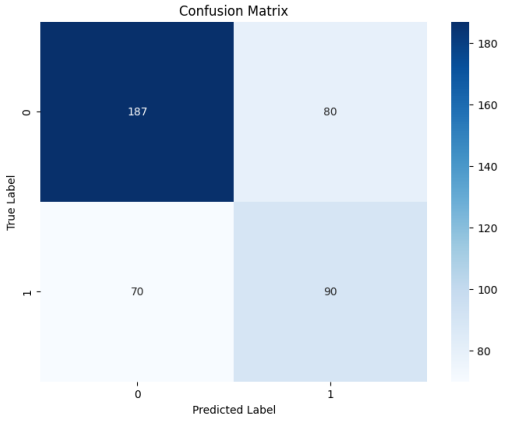
Tabel 6. 14 Hasil Evaluasi Learning Rate untuk Label 2 Kelas dari Epoch Terbaik pada Skenario Freeze Layer 0 hingga 3

Learning Rate	Accuracy	Recall	Precision	F1-Score
2e-5	0,6698	0,6698	0,6663	0,6677
3e-5	0,6487	0,6487	0,6534	0,6507

Tabel 6.14 merupakan daftar performa model saat *freeze encoder layer* 0 hingga 3 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*.



Gambar 6. 3 Confusion Matrix Skenario Label 2 Kelas dengan Learning Rate 2e-5 dan Freeze Layer 0 hingga 3



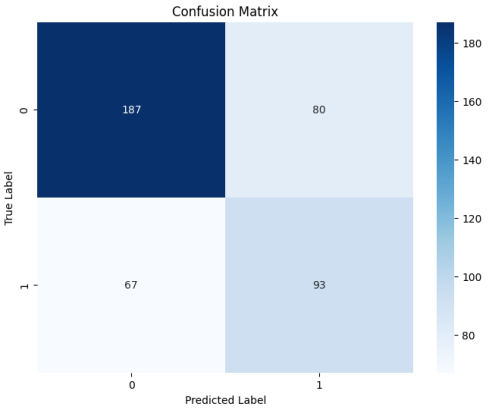
Gambar 6. 4 Confusion Matrix Skenario Label 2 Kelas dengan Learning Rate 3e-5 dan Freeze Layer 0 hingga 3

Performa terbaik ditunjukkan saat *learning rate* 2e-5 dimana *accuracy* dan *recall* cukup tinggi sebesar 66,98% yang berarti model cukup baik mengenali label sebenarnya sedangkan untuk *learning rate* 3e-5 *accuracy* dan *recall* tidak berbeda signifikan yaitu 64,87% . Dari metriks lainnya yaitu *precision* dan *F1-score* yang terbaik menggunakan *learning rate* 2e-5 dimana *precision* hanya berbeda sedikit dari *recall* yaitu 66,63% yang berarti model hanya sedikit melakukan salah prediksi yang berakibat pada *F1-score* yang tinggi yaitu 66,77% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada *learning rate* 3e-5 *precision* lebih tinggi dari *recall* yaitu 65,34% dimana model lebih jarang melakukan kesalahan prediksi sehingga *F1-score* menjadi tinggi 65,07% yang berarti model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada *learning rate* 3e-5 terlihat dari *confussion matrix* model mengalami *underfitting* dengan hanya memberikan prediksi pada salah satu kelas. Berdasarkan *confussion matrix* terlihat model lebih akurat memprediksi kelas *False* mayoritas saat *learning rate* 2e-5 tapi lebih akurat memprediksi kelas *True* saat *learning rate* 3e-5.

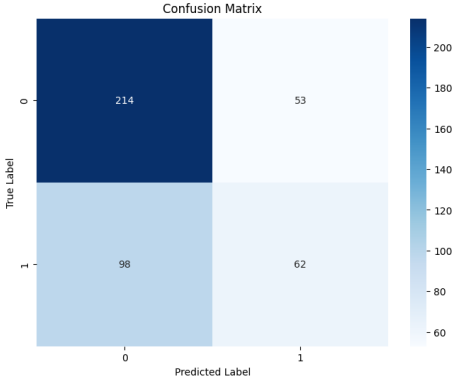
Tabel 6. 15 Hasil Evaluasi Learning Rate untuk Label 2 Kelas dari Epoch Terbaik pada Skenario Freeze Layer 0 hingga 5

Learning Rate	Accuracy	Recall	Precision	F1-Score
2e-5	0,6557	0,6557	0,6618	0,6582
3e-5	0,6464	0,6464	0,6309	0,6312

Tabel 6.15 merupakan daftar performa model saat freeze encoder layer 0 hingga 5 yang diukur menggunakan metriks klasifikasi accuracy, recall, precision, dan F1-score.



Gambar 6. 5 Confusion Matrix Skenario Label 2 Kelas dengan Learning Rate 2e-5 dan Freeze Layer 0 hingga 5



Gambar 6. 6 Confusion Matrix Skenario Label 2 Kelas dengan Learning Rate 3e-5 dan Freeze Layer 0 hingga 5

Performa terbaik ditunjukkan saat learning rate 2e-5 dimana *accuracy* dan *recall* cukup tinggi sebesar 65,57% yang berarti model cukup baik mengenali label sebenarnya sedangkan untuk *learning rate* 3e-5 *accuracy* dan *recall* tidak berbeda signifikan

yaitu 64,64%. Dari metriks lainnya yaitu *precision* dan *F1-score* yang terbaik menggunakan *learning rate* 2e-5 dimana *precision* lebih baik dari *recall* yaitu 66,18% yang berarti model hanya sedikit melakukan salah prediksi yang berakibat pada *F1-score* yang tinggi yaitu 65,82% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada *learning rate* 3e-5 *precision* lebih tinggi dari *recall* yaitu 63,09% dimana model lebih jarang melakukan kesalahan prediksi sehingga *F1-score* menjadi tinggi 63,12% yang berarti model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Berdasarkan *confussion matrix* terlihat model lebih akurat memprediksi kelas *False* mayoritas saat *learning rate* 3e-5 tapi lebih akurat memprediksi kelas *True* saat *learning rate* 2e-5.

Tabel 6. 16 Hasil Evaluasi Freeze Layer untuk Label 2 Kelas dari Epoch Terbaik pada Skenario Learning Rate 2e-5

Layer Freeze	Accuracy	Recall	Precision	F1-Score
0 – 1	0,6651	0,6651	0,6594	0,6614
0 – 3	0,6698	0,6698	0,6663	0,6677
0 – 5	0,6557	0,6557	0,6618	0,6582

Tabel 6.16 merupakan daftar performa model saat *learning rate* 2e-5 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*.

Performa terbaik ketika model *freeze layer* 0 hingga 3 dimana accuracy dan recall sebesar 66,98% yang berarti model cukup baik mengenali label sebenarnya sedangkan lainnya tidak berbeda jauh dari segi *accuracy* dan *recall*, dimana *accuracy* dan *recall freeze layer* 0 hingga 1 paling mendekati sebesar 66,51% sedangkan paling rendah pun hanya berbeda sedikit yaitu 65,57%. Pada saat *freeze layer* 0 hingga 3 *precision* dan *F1-score* hanya berbeda sedikit dari *recall* yaitu 66,63% yang berarti model hanya sedikit melakukan salah prediksi yang berakibat pada *F1-score* yang

tinggi yaitu 66,77% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada saat freeze layer 0 hingga 1 *precision* dan *F1-score* hanya berbeda sedikit dari *recall* yaitu 65,94% yang berarti model hanya sedikit melakukan salah prediksi yang berakibat pada *F1-score* yang tinggi yaitu 66,14% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada saat *freeze layer* 0 hingga 1 *precision* dan *F1-score* lebih tinggi dari *recall* yaitu 66,18% dimana model lebih jarang melakukan kesalahan prediksi sehingga *F1-score* menjadi tinggi yaitu 65,82% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Dari *confussion matrix* Gambar 6.1 dan 6.3 terlihat model lebih akurat memprediksi kelas False dengan 201 dan 203 sedangkan Gambar 6.5 memperlihatkan model lebih akurat memprediksi kelas True dengan 93.

Tabel 6. 17 Hasil Evaluasi Freeze Layer untuk Label 2 Kelas dari Epoch Terbaik pada Skenario Learning Rate 3e-5

Layer Freeze	Accuracy	Recall	Precision	F1-Score
0 – 1	0,6253	0,6253	0,391	0,4811
0 – 3	0,6487	0,6487	0,6534	0,6507
0 – 5	0,6464	0,6464	0,6309	0,6312

Tabel 6.17 merupakan daftar performa model saat *learning rate* 3e-5 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*.

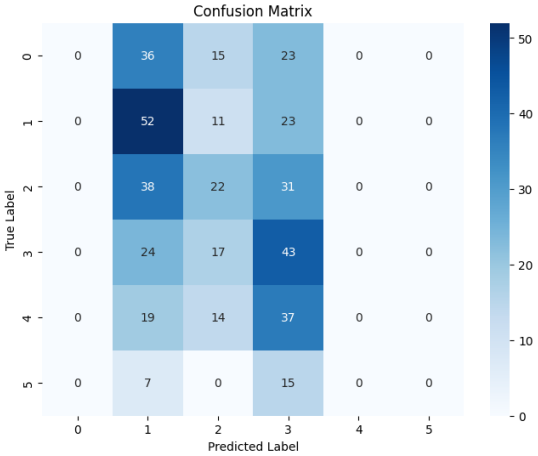
Performa terbaik ketika model *freeze layer* 0 hingga 3 dimana *accuracy dan recall* sebesar 64,87% yang berarti model cukup baik mengenali label sebenarnya sedangkan lainnya tidak berbeda jauh dari segi *accuracy* dan *recall*, dimana *accuracy* dan *recall freeze layer* 0 hingga 5 paling mendekati sebesar 64,64% sedangkan paling rendah pun tidak berbeda signifikan yaitu 62,53%. Pada saat *freeze layer* 0 hingga 3 *precision* lebih tinggi dari *recall* yaitu

65,34% dimana model lebih jarang melakukan kesalahan prediksi sehingga *F1-score* menjadi tinggi yaitu 65,07% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada saat *freeze layer* 0 hingga 5 *precision* dan *F1-score* hanya berbeda sedikit dari recall yaitu 63,09% yang berarti model hanya sedikit melakukan salah prediksi yang berakibat pada *F1-score* yang tinggi yaitu 63,12% dimana model seimbang dalam memberikan prediksi dan juga mengenali label sebenarnya. Pada saat *freeze layer* 0 hingga 1 *precision* berbeda signifikan dengan recall yaitu 39,1% dimana model kurang akurat melakukan prediksi sehingga *F1-score* menjadi rendah yaitu 48,11% dimana tidak seimbang hanya baik pada satu aspek yaitu menangkap label sebenarnya tapi kurang dalam memprediksi kelas. Dari *confussion matrix* Gambar 6.6 terlihat model lebih akurat memprediksi kelas *False* dengan 214 sedangkan Gambar 6.4 memperlihatkan model lebih akurat memprediksi kelas *True* dengan 90.

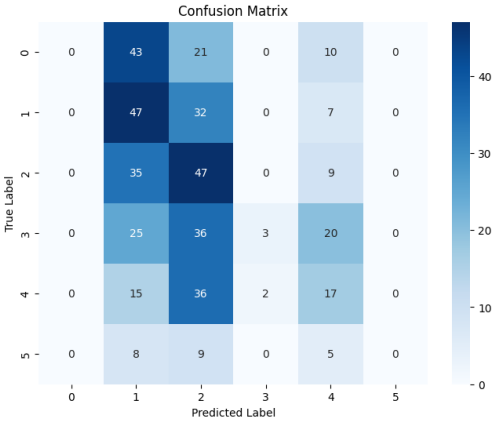
Tabel 6. 18 Hasil Evaluasi Learning Rate untuk Label 6 Kelas dari Epoch Terbaik pada Skenario Freeze Layer 0 hingga 1

Learning Rate	Accuracy	Recall	Precision	F1-Score
2e-5	0,274	0,274	0,168	0,2012
3e-5	0,267	0,267	0,2691	0,2004

Tabel 6.18 merupakan daftar performa model saat *freeze encoder layer* 0 hingga 1 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*



Gambar 6. 7 Confusion Matrix Skenario Label 6 Kelas dengan Learning Rate 2e-5 dan Freeze Layer 0 hingga 1



Gambar 6. 8 Confusion Matrix Skenario Label 6 Kelas dengan Learning Rate 3e-5 dan Freeze Layer 0 hingga 1

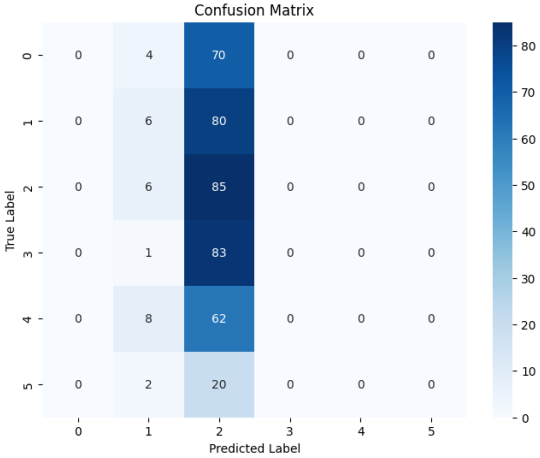
Performa terbaik saat *learning rate* 2e-5 dimana *accuracy* dan *recall* sebesar 27,4% yang hanya berbeda sedikit dengan *learning*

rate 3e-5 sebesar 26,7% yang berarti model cukup baik mengenali label sebenarnya dari berita. Meskipun begitu dari segi *precision learning rate* 3e-5 bahkan lebih baik sedikit dari recall sebesar 26,91% sedangkan saat *learning rate* 2e-5 *precision* jatuh menjadi 16,1% yang berarti saat 2e-5 model kesulitan memprediksi label yang termasuk ke kelas tersebut. Dari *F1-score* keduanya tidak berbeda jauh yaitu 20,12% dengan 20,04% yang berarti keduanya cukup seimbang dalam memprediksi label yang termasuk ke kelas dan juga mengenali label sebenarnya.

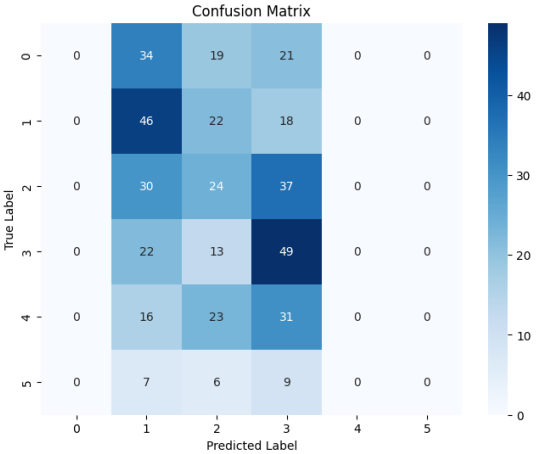
Tabel 6. 19 Hasil Evaluasi Learning Rate untuk Label 6 Kelas dari Epoch Terbaik pada Skenario Freeze Layer 0 hingga 3

Learning Rate	Accuracy	Recall	Precision	F1-Score
2e-5	0,2131	0,2131	0,09	0,0952
3e-5	0,2787	0,2787	0,166	0,206

Tabel 6.19 merupakan daftar performa model saat *freeze encoder layer* 0 hingga 3 yang diukur menggunakan metriks klasifikasi *accuracy*, *recall, precision*, dan *F1-score*



Gambar 6. 9 Confusion Matrix Skenario Label 6 Kelas dengan Learning Rate 2e-5 dan Freeze Layer 0 hingga 3



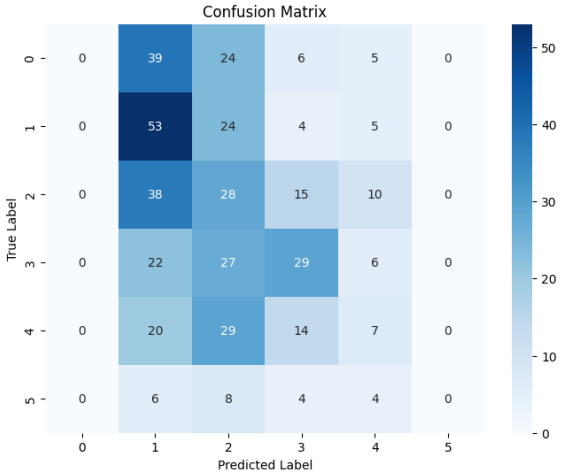
Gambar 6. 10 Confusion Matrix Skenario Label 6 Kelas dengan Learning Rate 3e-5 dan Freeze Layer 0 hingga 3

Performa terbaik saat *learning rate* 3e-5 dimana *accuracy* dan *recall* sebesar 27,87% yang berbeda cukup signifikan dengan *learning rate* 2e-5 yang hanya 21,31% yang berarti model cukup baik mengenali label sebenarnya dari berita saat *learning rate* 3e-5. Meskipun begitu dari segi *precision* keduanya jatuh dimana saat *learning rate* 16,6% sedangkan *learning rate* 2e-5 lebih rendah lagi dengan hanya 9% yang berarti keduanya kesulitan memprediksi label yang termasuk ke kelas tersebut. Dari *F1-score learning rate* 3e-5 masih cukup dengan 20,6% yang berarti model masih cukup seimbang sedangkan saat *learning rate* 2e-5 hanya 9,52% yang dapat disebabkan oleh *precision* yang sangat rendah.

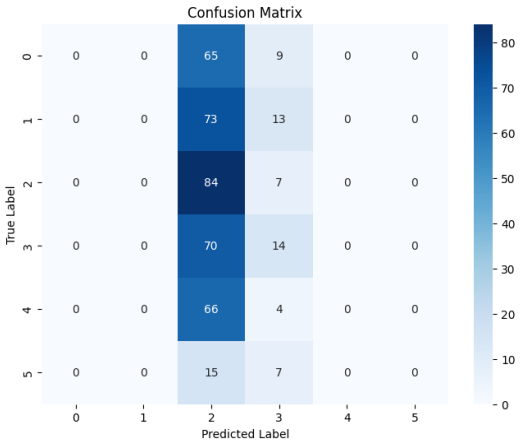
Tabel 6. 20 Hasil Evaluasi Learning Rate untuk Label 6 Kelas dari Epoch Terbaik pada Skenario Freeze Layer 0 hingga 5

Learning Rate	Accuracy	Recall	Precision	F1-Score
2e-5	0,274	0,274	0,2128	0,2271
3e-5	0,2295	0,2295	0,099	0,1171

Tabel 6.20 merupakan daftar performa model pada 6 kelas saat *freeze encoder layer* 0 hingga 5 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*



Gambar 6. 11 Confusion Matrix Skenario Label 6 Kelas dengan Learning Rate 2e-5 dan Freeze Layer 0 hingga 5



Gambar 6. 12 Confusion Matrix Skenario Label 6 Kelas dengan Learning Rate 3e-5 dan Freeze Layer 0 hingga 5

Performa terbaik saat *learning rate* 2e-5 dimana *accuracy* dan *recall* sebesar 27,4% yang berbeda cukup signifikan dengan *learning rate* 3e-5 yang hanya 22,95% yang berarti model cukup baik mengenali label sebenarnya dari berita saat *learning rate* 2e-5. Meskipun begitu dari segi *precision* saat *learning rate* 2e-5 berbeda cukup signifikan dengan 21,28% sedangkan *learning rate* 2e-5 jauh lebih rendah lagi dengan hanya 9,9% yang berarti saat *learning rate 3e-5* kesulitan memprediksi label yang termasuk ke kelas tersebut. Dari *F1-score learning rate* 2e-5 masih cukup tinggi dengan 22,71% yang berarti model masih cukup seimbang sedangkan saat *learning rate* 3e-5 hanya 11,71% yang dapat disebabkan oleh *precision* yang sangat rendah.

Tabel 6. 21 Hasil Evaluasi Freeze Layer untuk Label 6 Kelas dari Epoch Terbaik pada Skenario Learning Rate 2e-5

Layer Freeze	Accuracy	Recall	Precision	F1-Score

0 – 1	0,274	0,274	0,168	0,2012
0 – 3	0,2131	0,2131	0,09	0,0952
0 – 5	0,274	0,274	0,2128	0,2271

Tabel 6.21 merupakan daftar performa model saat *learning rate* 2e-5 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*.
Performa terbaik ketika model *freeze layer* 0 hingga 5 dari keseluruhan metriks. Dari segi *accuracy* dan *recall* sama dengan *freeze layer* 0 hingga 1 yaitu 27,4% tapi lebih baik dari segi *precision* dan *F1-score* yaitu 21,28% dan 22,71% yang berarti keduanya baik dalam mengenali label sebenarnya tapi *freeze layer* 0 hingga 5 cukup lebih unggul dalam memprediksi yang termasuk ke kelas tersebut. Pada *freeze layer* 0 hingga 3 memiliki perbedaan *accuracy* dan *recall* yang cukup signifikan yaitu yang hanya 21,31% yang berarti masih kurang dalam mengenali label sebenarnya dibanding yang lain. Selain itu, precision dan F1-score lebih buruk lagi dengan hanya 9% dan 9,52% yang berarti model buruk dalam memprediksi label yang masuk ke dalam kelas tersebut.

Tabel 6. 22 Hasil Evaluasi Freeze Layer untuk Label 6 Kelas dari Epoch Terbaik pada Skenario Learning Rate 3e-5

Layer Freeze	Accuracy	Recall	Precision	F1-Score
0 – 1	0,267	0,267	0,2691	0,2004
0 – 3	0,2787	0,2787	0,166	0,206
0 – 5	0,2295	0,2295	0,099	0,1171

Tabel 6.22 merupakan daftar performa model saat *learning rate* 3e-5 yang diukur menggunakan metriks klasifikasi *accuracy, recall, precision*, dan *F1-score*

Performa terbaik ketika model *freeze layer* 0 hingga 3 dari segi *accuracy* dan *recall* sedangkan *freeze layer* 0 hingga 1 mendekati dengan 26,7% yang berarti model cukup baik mengenali label sebenarnya. Pada *freeze layer* 0 hingga 3 *precision* berbeda cukup signifikan yaitu 16,6% sehingga *F1-score* hanya 20,6% yang berarti model kurang baik dalam memprediksi label dari kelas tersebut tapi cukup seimbang antara *recall* dan *precision*. Pada *freeze layer* 0 hingga 1 memiliki *precision* yang lebih tinggi dari *recall* yaitu 26,91% tapi hanya memiliki *F1-score* 20,04% yang berarti model baik dalam memprediksi label dari kelas tersebut tapi cukup seimbang antara *recall* dan *precision*. Pada *freeze layer* 0 hingga 5 *accuracy* dan *recall* cukup rendah jika dibandingkan yang lain yaitu 22,95%. Dari segi *precision* jauh lebih rendah dari *recall* yaitu 9,9% sehingga berdampak pada *F1-score* yang rendah hanya 11,71% dimana tidak seimbang antara *precision* dan *recall*.

Untuk skenario label 2 kelas penggunaan *learning rate* 2e-5 dan *freeze layer* 0 hingga 3 selalu mendapat performa terbaik dimana *accuracy* dan *recall* sebesar 66,98% dengan *precision* 66,63% dan *F1-score* 66,77%. Dengan tidak melatih layer awal seperti layer 0 hingga 3 yang berfungsi untuk representasi sintaksis dasar dapat mencegah kemungkinan *overfitting* dan mempercepat konvergensi. *Learning rate* 2e-5 dari eksperimen menunjukkan memungkinkan pelatihan model yang stabil dengan tetap adaptif menangkap pola semantik pada berita. Untuk skenario label 6 kelas jauh lebih buruk daripada label 2 kelas dimana memprediksi ke sejumlah yang paling banyak hanya 4 kelas terkadang mayoritas hanya ke salah satu kelas. Performa terbaik dari segi *accuracy* dan *recall* saat *learning rate* 3e-5 dan *freeze layer* 0 hingga 3 sebesar 27,87% sedangkan *precision* hanya 16,6%. Meskipun tidak ada performa yang benar-benar konsisten tapi dari pengujian *learning rate* hasil *freeze layer* 0 hingga 1 selalu antara yang terbaik atau mendekati terbaik selain itu dari kedua *learning rate* mayoritas mendapat hasil lebih tinggi saat *learning rate* 3e-5. Hal tersebut menjadi perkiraan kasar untuk skenario terbaik dari label 6 kelas

adalah menggunakan *learning rate* 3e-5 dan *freeze layer* 0 hingga 1 meskipun hasil tidak optimal jika dibandingkan dengan label 2 kelas.

Tabel 6. 23 Tabel Perbandingan antar Skenario

Skenario	Pembahasan
Freeze Encoder Layer	Freeze encoder layer 0 – 1 optimal untuk 6 kelas karena dengan tugas lebih kompleks membutuhkan lebih banyak tuning agar model lebih adaptif ke pola-pola spesifik pada tugas baru. Freeze encoder layer 0 – 3 optimal untuk 2 kelas karena relatif tidak se-kompleks dengan 6 kelas dimana masih mempertahankan bobot untuk representasi umum dari pre-trained layer awal yang dilatih pada korpus yang cukup besar.
Learning Rate	Learning rate 2e-5 optimal untuk 2 kelas dapat disebabkan untuk menjaga update dari gradient tetap stabil untuk mencegah overfitting yang lebih mungkin terjadi ketika tugas tidak terlalu kompleks. Learning rate 3e-5 optimal untuk 6 kelas dapat disebabkan untuk update gradient perlu lebih agresif agar menemukan pola yang tepat dari eksplorasi representasi baru untuk klasifikasi yang lebih rumit.
Jumlah Kelas Klasifikasi	Dengan 2 label kelas memberikan hasil metriks klasifikasi yang jauh lebih tinggi karena kompleksitas konteks yang dibutuhkan model lebih rendah dibanding 6 label kelas meskipun 2 label kelas memiliki kelas yang imbalance.

[Halaman ini sengaja dikosongkan]

BAB VII
KESIMPULAN DAN SARAN

7.1. Kesimpulan

Tahapan kerja praktik ini diawali dengan *preprocessing* untuk mempersiapkan data agar dapat digunakan sebagai input model sehingga menghasilkan prediksi yang optimal. Tahap *preprocessing* dengan membuat *metadata* dari fitur yang dianggap relevan agar dapat memberi informasi tambahan bagi *classifier* model malalui makna semantik dari model BERT. Penggunaan BERT Tokenizer untuk mengubah kalimat pada berita menjadi *sub word* memudahkan model fleksibel merepresentasikan *weight attention* setiap id *token* sebagai makna semantik. Selama training BERT model tidak melatih semua *encoder layer* mempercepat konvergensi dan menghemat waktu komputasi. Hasil *weight* setiap *token* dari model BERT menjadi input bagi model Bi-LSTM yang dapat menangkap konteks dari 2 arah dapat mempertahankan konteks dari BERT dan terakhir *classifier* untuk mengklasifikasikan input semantik tersebut ke dalam label kelas.

Hasil uji coba menunjukkan model memberi performa cukup optimal untuk label 2 kelas tapi kesulitan jika terdapat 6 kelas. Saat skenario 6 kelas model cenderung hanya memprediksi ke salah satu kelas sehingga performa cenderung buruk. Dari segi performa skenario label 2 kelas menggunakan *learning rate* 2e-5 dan *freeze layer* 0 hingga 3 selain menunjukkan *accuracy* yang baik tapi juga seimbang antara *recall* dan *precision*. Penggunaan *learning rate* 2e-5 cenderung optimal karena menjaga stabilitas parameter *pretrained* model untuk belajar task baru. Alasan tidak melatih semua *encoder layer* dari BERT untuk memanfaatkan bobot dari *pretrained* model yang mayoritas untuk struktur kalimat dan arti semantik di layer awal seperti layer pertama hingga keempat sehingga juga mengurangi kemungkinan *overfitting* model ke data training.

7.2. Saran

Saran implementasi model klasifikasi teks menggunakan representasi vektor kata adalah eksplorasi metode representasi vektor kata dinamis lainnya seperti RoBERTa.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] A. Khanifah and A. M. Fauzi, "Dampak Berita Hoax Tentang Covid-19 Terhadap Pelaksanaan Protokol Kesehatan Oleh Masyarakat," *J. Ilm. Din. Sos.*, vol. 6, no. 2, pp. 250–267, 2022, doi: 10.38043/jids.v6i2.3485.

[2] D. A. N. Krisna and U. Salamah, "Perbandingan Algoritma Naïve Bayes Dan K-Nearest Neighbor Untuk Klasifikasi Berita Hoax Kesehatan Di Media Sosial Twitter," *J. Tek. Inform. Kaputama*, vol. 6, no. 2, pp. 836–845, 2022.

[3] N. Rai, D. Kumar, N. Kaushik, C. Raj, and A. Ali, "Fake News Classification using transformer based enhanced LSTM and BERT," *Int. J. Cogn. Comput. Eng.*, vol. 3, no. March, pp. 98–105, 2022, doi: 10.1016/j.ijcce.2022.03.003.

[4] R. Anggrainingsih, G. M. Hassan, and A. Datta, "BERT based classification system for detecting rumours on Twitter," pp. 1–10, 2021, [Online]. Available: http://arxiv.org/abs/2109.02975

[5] "Sejarah." [Online]. Available: https://www.its.ac.id/informatika/id/tentang-kami/sejarah/

[6] "Visi dan Misi." [Online]. Available: https://www.its.ac.id/informatika/id/tentang-kami/visi-dan-misi/

[7] "Struktur Organisasi." [Online]. Available: https://www.its.ac.id/informatika/id/dosen-staff/struktur-organisasi/

[8] “Laboratorium.” [Online]. Available: https://www.its.ac.id/informatika/id/fasilitas/laboratorium/

[9] M. Prabu, L. B. Thinesh, and M. Sreekumar, “Enhanced Fake News Detection Leveraging a Hybrid BERT-BiLSTM Model,” *2024 Glob. Conf. Commun. Inf. Technol. GCCIT 2024*, pp. 1–6, 2024, doi: 10.1109/GCCIT63234.2024.10862089.

[10] L. B. Angizeh and M. R. Keyvanpour, “Detecting Fake News using Advanced Language Models: BERT and RoBERTa,” *2024 10th Int. Conf. Web Res. ICWR 2024*, pp. 46–52, 2024, doi: 10.1109/ICWR61162.2024.10533352.

[11] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” no. 1, 2019, [Online]. Available: http://arxiv.org/abs/1907.11692

[12] E. G. Figueiredo, U. P. Clival, B. A. A. Rtery, N. R. Crawford, P. D, and E. T. Al, “C Omparative a Nalysis of a Nterior P Etrosectomy,” *Neurosurgery*, vol. 58, no. February, pp. 13–21, 2006.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc.*, pp. 1–12, 2013.

[14] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, 2017, doi: 10.1162/tacl_a_00051.

[15] J. Pennington, R. Socher, and C. D. Manning,

"GloVe: Global vectors for word representation," *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, no. June 2018, pp. 1532–1543, 2014, doi: 10.3115/v1/d14-1162.

[16] A. Vaswani, "Attention Is All You Need," no. Nips, 2017.

[17] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Naacl-Hlt 2019*, no. Mlm, pp. 4171–4186, 2018, [Online]. Available: https://aclanthology.org/N19-1423.pdf

[18] Z. Zhu and K. Mao, "Knowledge-based BERT word embedding fine-tuning for emotion recognition," *Neurocomputing*, vol. 552, 2023, doi: 10.1016/j.neucom.2023.126488.

[19] S. Shaheen, W. El-Hajj, H. Hajj, and S. Elbassuoni, "Emotion recognition from text based on automatically generated rules," *IEEE Int. Conf. Data Min. Work. ICDMW*, vol. 2015-Janua, no. January, pp. 383–392, 2015, doi: 10.1109/ICDMW.2014.80.

[20] K. Jia, F. Meng, J. Liang, and P. Gong, "Text sentiment analysis based on BERT-CBLBGA," *Comput. Electr. Eng.*, vol. 112, no. November, p. 109019, 2023, doi: 10.1016/j.compeleceng.2023.109019.

[21] M. Pota, M. Ventura, H. Fujita, and M. Esposito, "Multilingual evaluation of pre-processing for BERT-based sentiment analysis of tweets," *Expert Syst. Appl.*, vol. 181, no. April, p. 115119, 2021, doi:

10.1016/j.eswa.2021.115119.

[22] A. Tamkin, T. Singh, D. Giovanardi, and N. Goodman, “Investigating transferability in pretrained language models,” *Find. Assoc. Comput. Linguist. Find. ACL EMNLP 2020*, pp. 1393–1401, 2020, doi: 10.18653/v1/2020.findings-emnlp.125.

[23] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, 2000, doi: 10.1162/089976600300015015.

[24] R. Ofori-Boateng, M. Aceves-Martins, C. Jayne, N. Wiratunga, and C. F. Moreno-Garcia, “Evaluation of Attention-Based LSTM and Bi-LSTM Networks For Abstract Text Classification in Systematic Literature Review Automation,” *Procedia Comput. Sci.*, vol. 222, pp. 137–146, 2023, doi: 10.1016/j.procs.2023.08.149.

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS I

Nama	: Sandhika Surya Ardyanto
Tempat, Tanggal Lahir	: Sukoharjo, 21 Agustus 2003
Jenis Kelamin	: Laki-Laki
Telepon	: +6287808212003
Email	: 5025211022@student.its.ac.id

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 8 (Delapan)