SKRIPSI

DETEKSI UJARAN KEBENCIAN PADA TEKS BAHASA INGGRIS MENGGUNAKAN TRANSFER LEARNING HATEBERT DAN PENERAPAN EXPLAINABLE AI BERUPA LIME

Konang Tyagazain Nirangkara
21/474140/PA/20468

PROGRAM STUDI SARJANA ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA
YOGYAKARTA
2025

HALAMAN PENGESAHAN

SKRIPSI (RISET AKADEMIK)

DETEKSI UJARAN KEBENCIAN PADA TEKS BAHASA INGGRIS MENGGUNAKAN TRANSFER LEARNING HATEBERT DAN PENERAPAN EXPLAINABLE AI BERUPA LIME

Telah dipersiapkan dan disusun oleh

Konang Tyagazain Nirangkara
21/474140/PA/20468

Telah dipertahankan di depan Tim Penguji
pada tanggal 04 Juni 2025

Susunan Tim Penguji

Dr. Lukman Heryawan, S.T., M.T.
Ketua Penguji

Novera Istiqomah, S.T., M.T., Ph.D.
Anggota Penguji

Prof. Dr.-Ing. Mhd. Reza M. I. Pulungan, S.Si., M.Sc.
Pembimbing

Mengetahui,
a.n. Dekan FMIPA UGM
Wakil Dekan Bidang Pendidikan, Pengajaran dan Kemahasiswaan

Prof. Drs. Roto, M.Eng., Ph.D.
NIP. 196711171993031020

PERNYATAAN PLAGIARISME

Saya, yang bertanda tangan di bawah ini:

Nama : Konang Tyagazain Nirangkara
NIM : 21/474140/PA/20468
Tahun masuk : 2021
Program : S1 Ilmu Komputer
Fakultas : Matematika dan Ilmu Pengetahuan Alam

Dengan ini saya menyatakan bahwa dalam dokumen skripsi ini, tidak ada bagian yang diambil dari karya ilmiah lain yang diajukan untuk memperoleh gelar akademik di institusi pendidikan tinggi mana pun. Selain itu, tidak ada karya atau pendapat yang ditulis atau diterbitkan oleh individu/institusi lain, kecuali yang dikutip secara tertulis dalam dokumen ini dan dirujuk sepenuhnya dalam daftar pustaka.

Oleh karena itu, saya menyatakan bahwa dokumen ilmiah ini bebas dari plagiarisme, dan jika di kemudian hari terbukti bahwa dokumen skripsi ini adalah hasil plagiarisme dari karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil dari karya penulis lain, saya bersedia menerima sanksi akademik dan/atau hukum yang berlaku.

Yogyakarta, 29 Mei 2025

Konang Tyagazain Nirangkara
21/474140/PA/20468

INTISARI

DETEKSI UJARAN KEBENCIAN PADA TEKS BAHASA INGGRIS MENGGUNAKAN TRANSFER LEARNING HATEBERT DAN PENERAPAN EXPLAINABLE AI BERUPA LIME

Oleh

Konang Tyagazain Nirangkara

21/474140/PA/20469

Keterbatasan metode berbasis kata kunci dan model machine learning tradisional dalam mendeteksi ujaran kebencian yang bersifat tersirat, sarkasme, atau istilah terselubung sering kali mengurangi akurasi sistem deteksi. Penelitian ini mengusulkan penerapan *transfer learning* pada HateBERT, model berbasis *transformer* yang telah dilatih dengan dataset ujaran kebencian, untuk meningkatkan pemahaman konteks dalam klasifikasi teks pada dataset OffensEval 2019. Dengan menerapkan pra-pemrosesan teks, tokenisasi, dan *fine-tuning* model, pendekatan ini memungkinkan model menangkap makna ujaran kebencian secara lebih efektif. Selain itu, penelitian ini mengintegrasikan Explainable AI (XAI) menggunakan LIME untuk menganalisis kontribusi setiap kata terhadap keputusan model, sehingga meningkatkan transparansi dan interpretabilitas model deteksi ujaran kebencian. Evaluasi dilakukan menggunakan metrik *accuracy*, *precision*, *recall*, *macro F1-score*, dan *POS F1-score*, serta analisis interpretabilitas model melalui hasil XAI berupa LIME. Setelah dilakukan *transfer learning*, model mencapai *macro F1-score* sebesar 0.7878, menunjukkan performa yang baik dalam klasifikasi ujaran kebencian. Implementasi LIME menghasilkan visualisasi kontribusi kata berupa *highlight* warna, yang membantu pengguna memahami faktor-faktor yang memengaruhi prediksi model. Namun, penjelasan yang diberikan bersifat lokal dan tidak sepenuhnya konsisten, sehingga tetap diperlukan peran manusia untuk melakukan interpretasi yang tepat atas hasil yang ditampilkan.

Kata Kunci: Deteksi ujaran kebencian, HateBERT, Transfer learning, Explainable AI, LIME, NLP

ABSTRACT

HATE SPEECH DETECTION IN ENGLISH TEXT USING HATEBERT TRANSFER LEARNING AND EXPLAINABLE AI IMPLEMENTATION USING LIME

By

Konang Tyagazain Nirangkara

21/474140/PA/20469

The limitations of keyword-based methods and traditional machine learning models in detecting implied hate speech, sarcasm, or veiled terms often reduce the accuracy of detection systems. This research proposes applying transfer learning to HateBERT, a transformer-based model that has been trained with hate speech datasets, to improve context understanding in text classification on the OffensEval 2019 dataset. By applying text pre-processing, tokenization, and model fine-tuning, this approach enables the model to capture the meaning of hate speech more effectively. In addition, this research integrates Explainable AI (XAI) using LIME to analyze the contribution of each word to the model's decision, thus improving the transparency and interpretability of the hate speech detection model. Evaluation was conducted using accuracy, precision, recall, macro F1-score, and POS F1-score metrics, as well as model interpretability analysis through XAI results in the form of LIME. After transfer learning, the model achieved a macro F1-score of 0.7878, indicating good performance in hate speech classification. The LIME implementation produces a visualization of word contributions in the form of color highlights, which helps users understand the factors that influence the model's predictions. However, the explanations provided are localized and not fully consistent, so a human is still required to make a proper interpretation of the results.

Keywords: Hate speech detection, HateBERT, Transfer learning, Explainable AI, LIME, NLP

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Deteksi ujaran kebencian telah menjadi topik penelitian yang penting dalam bidang *Natural Language Processing* (NLP), terutama dengan meningkatnya komunikasi *online* dan penggunaan media sosial. Ujaran kebencian umumnya didefinisikan sebagai segala bentuk ekspresi yang merendahkan, menghina, atau mendiskriminasi individu atau kelompok berdasarkan atribut seperti ras, etnis, agama, gender, atau orientasi seksual (Mudde dan Rovira Kaltwasser, 2017). Meningkatnya frekuensi ujaran kebencian dalam platform digital menimbulkan risiko yang signifikan bagi keharmonisan sosial dan keamanan publik, sehingga dibutuhkan sistem deteksi otomatis yang efektif (Schmidt dan Wiegand, 2017).

Di Indonesia, urgensi deteksi ujaran kebencian semakin diperkuat dengan adanya regulasi hukum yang mengatur penyebaran ujaran kebencian. Salah satu pedoman hukum yang berlaku adalah Surat Edaran Kepala Kepolisian Negara Republik Indonesia No. SE/6/X/2015, yang bertujuan untuk mencegah penyebaran ujaran kebencian yang dapat menimbulkan diskriminasi, kekerasan, bahkan konflik sosial. SE tersebut mendefinisikan ujaran kebencian sebagai tindak pidana yang dapat berbentuk penghinaan, perbuatan tidak menyenangkan, provokasi, penghasutan, serta penyebaran berita bohong yang berpotensi memicu perpecahan di masyarakat. Lebih lanjut, SE ini juga mengatur berbagai media yang dapat digunakan dalam penyebaran ujaran kebencian, termasuk kampanye, media cetak, media sosial, ceramah keagamaan, dan lain sebagainya (Kepolisian Negara Republik Indonesia, 2015). Dengan adanya regulasi ini, penerapan teknologi deteksi ujaran kebencian menjadi semakin penting untuk membantu penegakan hukum dalam mengidentifikasi dan mencegah penyebaran ujaran kebencian secara.

Kompleksitas dalam mendeteksi ujaran kebencian muncul karena sifat bahasa yang kontekstual dan sering kali bersifat tersirat. Ujaran kebencian dapat disampaikan dalam berbagai bentuk, seperti penghinaan eksplisit, istilah terselubung, atau bahkan sarkasme, yang membuat sistem berbasis *rule-based* sulit mengenalinya (Davidson dkk., 2017). Fortuna dan Nunes (2018) mengategorikan

metode deteksi ujaran kebencian ke dalam tiga pendekatan utama, yaitu sistem *rule-based*, *machine learning*, dan *deep learning*. Sistem *rule-based* menggunakan daftar kata atau frasa yang telah ditentukan untuk mendeteksi ujaran kebencian, tetapi memiliki kelemahan dalam menangkap variasi bahasa dan makna kontekstual. Sebaliknya, metode berbasis *machine learning*, terutama Support Vector System (SVM) dan Naïve Bayes, mampu mengenali pola dalam data, tetapi masih menghadapi tantangan dalam menangani ujaran kebencian yang lebih kompleks.

Kemajuan dalam *deep learning* telah menghasilkan model berbasis *transformer*, seperti *Bidirectional Encoder Representations from Transformers* (BERT), yang telah menunjukkan keberhasilan dalam berbagai tugas NLP (Devlin dkk., 2019). *Transfer learning* dengan model BERT memungkinkan sistem untuk memanfaatkan pengetahuan dari dataset besar dan menyesuaikannya dengan dataset spesifik dalam area tertentu. Teknik ini efektif dalam mengatasi keterbatasan dataset berlabel yang terbatas untuk deteksi ujaran kebencian (Murel dan Kavlakoglu, 2024). Salah satu pengembangan model yang didesain khusus untuk tugas ini adalah HateBERT, sebuah model berbasis BERT yang telah dilatih pada data ujaran kebencian dari platform seperti Reddit dan Twitter. Caselli dkk., (2021) menunjukkan bahwa HateBERT memiliki performa yang lebih baik dibandingkan model standar dalam mendeteksi ujaran kebencian, terutama dalam memahami bahasa kasar yang sering ditemukan dalam media sosial. Istilah model standar di sini merujuk pada BERT asli, yang dilatih pada korpus data umum seperti Wikipedia dan BookCorpus (Devlin dkk., 2019), yang tidak secara eksplisit mencakup ujaran kebencian. Dengan demikian, model standar cenderung kesulitan memahami konteks ujaran kebencian yang tersirat atau bersifat slang.

Meskipun model berbasis *ensemble*, seperti kombinasi BERT, CNN, dan LSTM (Mnassri dkk., 2022), atau *hybrid* seperti gabungan CNN dan Bi-LSTM (Kibriya dkk., 2024) telah terbukti mampu mencapai akurasi yang lebih tinggi, pendekatan-pendekatan tersebut membutuhkan sumber daya komputasi yang jauh lebih besar. Dalam konteks penelitian ini, HateBERT dipilih karena menawarkan perbandingan yang baik antara efisiensi komputasi dan akurasi yang cukup,

sehingga memadai sebagai platform eksperimen untuk penerapan Explainable AI (XAI). Dengan memanfaatkan model yang relatif ringan ini, proses interpretasi dapat dijalankan secara praktis tanpa memerlukan infrastruktur yang kompleks, yang sesuai dengan tujuan penelitian sebagai studi eksploratif mengenai integrasi XAI.

Selain itu, pendekatan *transfer learning* yang diterapkan pada HateBERT memungkinkan model untuk tetap bekerja secara efektif meskipun dataset yang digunakan memiliki distribusi kelas yang tidak seimbang, sebagaimana terjadi pada dataset OffensEval 2019. Penggunaan *macro F1-score* sebagai metrik evaluasi membantu memastikan bahwa performa model tidak bias terhadap kelas mayoritas. Arsitektur *transformer* memungkinkan model untuk mempelajari representasi yang baik bagi kedua kelas, sehingga ketidakseimbangan data tidak secara signifikan mengurangi kemampuan model dalam mendeteksi kelas minoritas.

Meskipun model berbasis *deep learning* menunjukkan performa yang tinggi dalam deteksi ujaran kebencian, tantangan besar yang dihadapi adalah kurangnya transparansi dalam pengambilan keputusan model (Buhrmester dkk., 2019). Oleh karena itu, pendekatan XAI dibutuhkan untuk meningkatkan interpretabilitas model. Di antara berbagai metode XAI, *Local Interpretable Model-agnostic Explanations* (LIME) dipilih dalam penelitian ini karena kesederhanaan implementasinya, efisiensi komputasi, serta kemampuannya memberikan penjelasan lokal yang intuitif dalam bentuk kontribusi kata (Ribeiro dkk., 2016). Dibandingkan dengan metode lain seperti SHAP, yang meskipun lebih akurat namun memerlukan biaya komputasi yang lebih tinggi, LIME memberikan keseimbangan yang sesuai antara interpretabilitas dan kepraktisan. Oleh sebab itu, LIME dipandang cukup memadai untuk meningkatkan transparansi sistem deteksi ujaran kebencian tanpa menambah beban komputasi yang signifikan.

Mengintegrasikan metode XAI dalam sistem deteksi ujaran kebencian tidak hanya meningkatkan transparansi, tetapi juga membantu dalam mendeteksi dan mengurangi bias dalam model AI. Dengan memahami bagaimana model membuat keputusan dan mengidentifikasi kata-kata yang paling berkontribusi terhadap

klasifikasi, *developer* dapat mengoptimalkan sistem dan memastikan bahwa hasil deteksi lebih adil dan dapat diandalkan (Kibriya dkk., 2024).

Dalam penelitian ini, pemilihan model HateBERT dilakukan bukan semata-mata untuk mengejar akurasi tertinggi, melainkan untuk memungkinkan eksperimen implementasi XAI yang efisien secara komputasi. Model-model dengan arsitektur yang lebih kompleks, seperti kombinasi BERT-CNN-LSTM (Mnassri dkk., 2022), CNN-BiLSTM dengan XAI (Kibriya dkk., 2024), maupun *ensemble learning*, memang mampu mencapai akurasi yang lebih tinggi, tetapi memerlukan sumber daya komputasi yang besar. Oleh karena itu, HateBERT dipilih sebagai model yang lebih ringan, yang memungkinkan penelitian ini berfokus pada penerapan *transfer learning* dengan dataset baru, sekaligus mengevaluasi potensi penggunaan metode LIME sebagai alat interpretasi yang praktis dalam deteksi ujaran kebencian.

Berdasarkan perkembangan-perkembangan tersebut, penelitian ini mengusulkan sistem deteksi ujaran kebencian menggunakan HateBERT dengan *transfer learning*, serta mengintegrasikan LIME sebagai metode XAI untuk meningkatkan transparansi dalam sistem deteksi. Dengan kombinasi pendekatan ini, penelitian ini bertujuan untuk meningkatkan akurasi deteksi ujaran kebencian sekaligus memastikan interpretabilitas model, sehingga dapat mendukung pengembangan sistem AI yang lebih bertanggung jawab dalam mengatasi tantangan ujaran kebencian di media sosial.

1.2 Rumusan Masalah

Metode deteksi ujaran kebencian berbasis *keyword* tradisional memiliki keterbatasan dalam mengenali ujaran kebencian yang bersifat tersirat, seperti sarkasme atau istilah terselubung, sehingga sering menghasilkan prediksi yang tidak akurat. Selain itu, pengujar kebencian juga memiliki berbagai cara dalam menghindari deteksi, seperti sengaja salah mengeja atau menggunakan sinonim, semakin memperumit tugas deteksi otomatis. Meskipun *transfer learning* dengan model seperti HateBERT telah menunjukkan potensi dalam menangani kompleksitas ini, model *deep learning* sering kali bersifat "*black box*" sehingga keputusan yang diambil sulit dipahami oleh pengguna dan peneliti.

Masalah ini juga ditambah dengan kurangnya integrasi XAI yang mampu menjelaskan proses pengambilan keputusan oleh model, sehingga sulit untuk mengevaluasi akurasi dan keandalan sistem, terutama dalam mengurangi bias. Oleh karena itu, diperlukan solusi yang tidak hanya meningkatkan akurasi deteksi ujaran kebencian tetapi juga memastikan transparansi dan interpretabilitas model.

1.3 Batasan Masalah

Adapun batasan masalah yang diterapkan dalam penelitian ini adalah:

1. Penelitian ini dilakukan dengan menggunakan Jupyter Notebook dan bahasa pemrograman Python sebagai lingkungan pengembangan.
2. Metrik evaluasi model deteksi ujaran kebencian adalah *macro F1-score*, sementara evaluasi metode XAI dilakukan berdasarkan sejauh mana hasil interpretasi dapat menjelaskan keputusan model.
3. Dataset yang digunakan dalam penelitian ini adalah dataset ujaran kebencian yang dikumpulkan dari *tweet* Twitter dan bersumber dari paper berjudul Predicting the Type and Target of Offensive Posts in Social Media (Zampieri dkk., 2019).

1.4 Tujuan Penelitian

Penelitian ini merupakan studi eksploratif yang bertujuan untuk mengevaluasi penerapan *transfer learning* dan interpretasi model berbasis XAI pada deteksi ujaran kebencian. Secara khusus, penelitian ini bertujuan untuk:

1. Menghitung performa model HateBERT dalam mendeteksi ujaran kebencian pada teks berbahasa Inggris setelah dilakukan *transfer learning* dengan data OffensEval 2019.
2. Membandingkan performa model berdasarkan variasi pra-pemrosesan yang digunakan serta membandingkan model setelah *transfer learning* dengan model awal sebelum pelatihan ulang.
3. Mengimplementasikan metode interpretasi model berbasis XAI berupa LIME untuk mengevaluasi kemampuannya dalam memberikan interpretabilitas tambahan terhadap keputusan model HateBERT dalam klasifikasi ujaran kebencian.

1.5 Manfaat Penelitian

Penelitian ini diharapkan mampu memberikan solusi yang lebih efektif dan transparan untuk mendeteksi ujaran kebencian melalui penerapan model HateBERT dengan teknik *transfer learning*. Pendekatan ini diharapkan mampu menangani kompleksitas pola bahasa, seperti ujaran kebencian tersirat, sarkasme, atau istilah terselubung, yang sering kali sulit dikenali oleh metode tradisional.

Selain itu, integrasi XAI memungkinkan interpretasi keputusan model secara lebih jelas, sehingga meningkatkan transparansi dan membantu mengidentifikasi potensi bias pada sistem deteksi ujaran kebencian.

Hasil penelitian ini mampu berkontribusi pada pengembangan sistem deteksi ujaran kebencian yang lebih akurat, dapat dipertanggungjawabkan, dan sesuai dengan tantangan nyata di sosial media. Penelitian ini juga dapat menjadi referensi untuk penelitian di masa depan yang fokus pada pengembangan model deteksi ujaran kebencian atau penerapan XAI dalam konteks lainnya.

1.6 Sistematika Penulisan

1. BAB I: PENDAHULUAN

   Bab ini berisi uraian latar belakang penelitian, rumusan masalah penelitian, tujuan penelitian, batasan masalah penelitian, metodologi penelitian, serta sistematika penulisan penelitian.

2. BAB II: TINJAUAN PUSTAKA

   Bab ini berisi tinjauan pustaka dari penelitian terdahulu mengenai berbagai pendekatan dalam deteksi ujaran kebencian, pengaplikasian model untuk deteksi atau klasifikasi ujaran kebencian, dan integrasi XAI pada model AI.

3. BAB III: LANDASAN TEORI

   Bab ini berisi teori-teori yang digunakan dalam penelitian mengenai *transfer learning*, model HateBERT, NLP, serta konsep XAI untuk meningkatkan transparansi model.

4. BAB IV: METODE PENELITIAN

   Bab ini berisi metode penelitian model deteksi ujaran kebencian menggunakan HateBERT yang meliputi proses pra-pemrosesan data,

pelatihan ulang model dengan teknik *transfer learning*, dan integrasi XAI untuk menjelaskan hasil keputusan model.

5. BAB V: JADWAL PENELITIAN

   Bab ini berisi tentang jadwal penelitian secara detail dan terurut mulai dari tinjauan pustaka hingga laporan presentasi penelitian.

BAB II

TINJAUAN PUSTAKA

2.1. Deteksi Ujaran Kebencian

Salah satu studi mendasar pada deteksi ujaran kebencian adalah oleh Fortuna dan Nunes. (2018) yang melakukan survei komprehensif tentang deteksi ujaran kebencian otomatis. Mereka meninjau metodologi yang ada dan mengklasifikasikannya ke dalam tiga kategori utama: pendekatan *rule-based*, teknik *machine learning*, dan model *deep learning*. Sistem *rule-based* menggunakan daftar kata atau frasa yang menyinggung yang telah ditentukan untuk mengidentifikasi ujaran kebencian, namun sering kali mengalami kesulitan dengan dinamika bahasa yang digunakan dalam forum *online*, sehingga menghasilkan *false-positive* yang tinggi. Sebaliknya, teknik *machine learning*, terutama yang menggunakan Support Vector Machines (SVM) dan Naïve Bayes *classifier*, menunjukkan kinerja yang lebih baik dengan *training* pada data berlabel, meskipun masih menghadapi tantangan dengan konteks dan perbedaan dalam ekspresi ujaran kebencian.

Fortuna dan Nunes menekankan bahwa meskipun teknik *machine learning* memberikan akurasi yang lebih baik daripada sistem *rule-based*, teknik ini sering kali membutuhkan rekayasa fitur yang kompleks. Proses ini melibatkan pemilihan fitur-fitur yang relevan secara manual dari data teks untuk meningkatkan kinerja model. Namun, banyak penelitian yang ada tidak memiliki dataset yang kuat untuk pelatihan dan evaluasi, yang membatasi kemampuan generalisasi temuan mereka. Penelitian ini menjadi dasar untuk perkembangan deteksi ujaran kebencian melalui teknik machine learning, yang melakukan ekstraksi fitur secara otomatis dan dapat menangkap pola yang lebih kompleks dalam teks.

Berdasarkan hasil tersebut, Mnassri dkk. (2022) mengusulkan sebuah metode lainnya dengan menggunakan metode *ensemble* yang dikombinasikan dengan BERT untuk mendeteksi ujaran kebencian. BERT adalah model berbasis *transformer* yang memproses teks secara dua arah, yang memungkinkannya untuk memahami konteks kata berdasarkan kata-kata di sekitarnya. Para peneliti berfokus pada pengintegrasian beberapa model *deep learning*, termasuk BERT,

Convolutional Neural Networks (CNN), dan *Long Short-Term Memory* (LSTM). Mereka membuat dataset yang seimbang dengan menggabungkan tiga dataset Twitter yang tersedia secara publik–Davidson, HatEval2019, dan OLID–yang menghasilkan dataset baru yang disebut DHO. Dataset ini memungkinkan *multi-label classification tweet* ke dalam kategori kebencian, menyinggung, atau netral.

Para penulis melaporkan bahwa model *ensemble* mereka mencapai F1-score sebesar 0.9278 pada dataset Davidson dengan menggunakan teknik *stacking*, yang menggabungkan prediksi beberapa model untuk meningkatkan kinerja secara keseluruhan. Mereka juga mencatat bahwa meskipun pendekatan *ensemble* membutuhkan lebih banyak sumber daya komputasi, pendekatan ini secara signifikan mengungguli model individu dalam hal akurasi klasifikasi. Studi ini menunjukkan keefektifan penggabungan arsitektur yang berbeda untuk meningkatkan kemampuan deteksi ujaran kebencian.

Berbeda dengan penelitian sebelumnya, Kibriya dkk. (2024) mengembangkan model dengan pendekatan *hybrid* yang mengintegrasikan teknik deep learning dan implementasi XAI. Mereka menggunakan kombinasi jaringan CNN dan jaringan *Bidirectional Long Short-Term Memory* (Bi-LSTM) yang dilatih dengan dataset 15.000 *tweet* yang diberi label untuk berbagai bentuk ujaran kebencian. CNN efektif untuk tugas klasifikasi teks karena kemampuannya untuk secara otomatis mempelajari fitur dari data teks mentah melalui lapisan konvolusi yang mendeteksi pola lokal. Jaringan Bi-LSTM meningkatkan kemampuan ini dengan memproses sekuens dua arah, memungkinkan mereka untuk menangkap konteks dari token sebelum dan sesudah input.

Model mereka mendapat *F1-score* 0.93, menunjukkan peningkatan signifikan dari model sebelumnya sekaligus menjelaskan bagaimana model pengambilan keputusan dari implementasi XAI. Peneliti menegaskan bahwa pendekatan yang mereka lakukan tidak hanya meningkatkan performa klasifikasi, namun juga menjawab salah satu tantangan penting dalam model AI yaitu transparansi.

Kibriya dkk. juga membahas hasil dari integrasi XAI dengan sistem pendeteksi ujaran kebencian. Mereka berpendapat bahwa kejelasan dari prediksi

model itu penting untuk menumbuhkan kepercayaan di antara user yang mengandalkan sistem otomatis untuk mengidentifikasi konten yang membahayakan. Temuan mereka menunjukkan bahwa pengguna lebih cenderung menerima dan bertindak berdasarkan prediksi model ketika mereka menerima penjelasan yang jelas tentang bagaimana prediksi tersebut dibuat.

Terakhir, Caselli dkk. (2021) memperkenalkan HateBERT, modifikasi dari BERT untuk mendeteksi bahasa kasar dalam Bahasa Inggris. Dengan melatih ulang BERT pada dataset yang ditargetkan yang berasal dari komentar Reddit yang tertandai sebagai komentar menyinggung yang dikenal sebagai RAL-E, mereka mencapai hasil dengan *F1-score* 0.809. HateBERT dibangun dengan mengubah arsitektur BERT tetapi difokuskan pada deteksi bahasa menghina dengan menyempurnakannya pada dataset tertentu.

Peneliti menekankan bahwa arsitektur HateBERT mampu menghasilkan model tradisional secara signifikan ketika diterapkan pada dataset seperti OffensEval dan HatEval. Mereka melakukan eksperimen yang membandingkan HateBERT dengan model dasar lainnya dan menemukan bahwa HateBERT secara konsisten memberikan kinerja yang unggul di berbagai metrik. Studi ini menunjukkan pentingnya *training* yang disesuaikan dengan kasus spesifik untuk meningkatkan kinerja model dalam mendeteksi ujaran kebencian.

Selain itu, Caselli dkk. meneliti dampak dari penyempurnaan BERT pada dataset spesifik yang disesuaikan dengan deteksi ujaran kebencian. Mereka menunjukkan bahwa *training* yang ditargetkan ini tidak hanya meningkatkan akurasi, tetapi juga memungkinkan model untuk lebih memahami konteks ujaran kebencian. Penelitian mereka menghasilkan standar baru untuk penelitian kedepannya di bidang ini dan menunjukkan potensi untuk memanfaatkan *transfer learning* untuk menangani kasus NLP yang kompleks seperti deteksi ujaran kebencian secara efektif.

Tabel 2.1 merangkum beberapa studi penelitian terbaru dalam deteksi ujaran kebencian dengan berbagai metode yang digunakan. Mnassri dkk. (2022) menggunakan pendekatan *ensemble learning* dengan kombinasi BERT, CNN, dan LSTM, yang berhasil mencapai *F1-score* 0.9278 dengan dataset DHO. Kibriya dkk.

(2024) mengimplementasikan CNN dan Bi-LSTM yang dikombinasikan dengan XAI pada dataset berisi 15.000 *tweet* berlabel, yang mendapat *F1-Score* 0.92. Sementara itu, Caselli dkk. (2021) memperkenalkan HateBERT, sebuah model *transformer* yang dioptimalkan dengan dataset RAL-E, mendapat *F1-Score* 0.809. Dari hasil studi ini, terlihat bahwa pendekatan berbasis *deep learning*, terutama yang dikombinasikan dengan *transfer learning*, dapat meningkatkan efektivitas deteksi ujaran kebencian.

Tabel 2.1 Perbandingan studi penelitian deteksi ujaran kebencian

Peneliti	Metode	Dataset	Akurasi/F1-Score
(Mnassri dkk., 2022)	Ensemble dengan BERT, CNN, LSTM	DHO (Davidson, HatEval2019, OLID)	F1-Score: 0.9278
(Kibriya dkk., 2024)	CNN dan Bi-LSTM dengan XAI	15.000 *tweet* berlabel	F1-Score: 0.93
(Caselli dkk., 2021)	HateBERT	RAL-E (Reddit comments)	F1-Score: 0.809

2.2. Explainable AI (XAI)

Salah satu kontribusi penting yang mendasari XAI dilakukan oleh Ribeiro dkk. (2016), yang mengembangkan LIME, sebuah teknik yang dirancang untuk menjelaskan prediksi model dengan cara yang dapat ditafsirkan dan dapat dipercaya. Penelitian mereka menjawab kebutuhan penting untuk memahami mengapa model *machine learning* membuat keputusan tertentu, terutama dalam skenario yang mengutamakan kepercayaan dan akuntabilitas. LIME bertujuan untuk memberikan penjelasan lokal dengan memperkirakan model yang kompleks dengan model yang lebih sederhana dan dapat ditafsirkan di sekitar prediksi tertentu.

Ribeiro dkk. mengusulkan untuk membuat penjelasan dengan melakukan modifikasi data input di input tertentu dan mengamati perubahan prediksi model. Mereka kemudian menggunakan modifikasi tersebut untuk melatih model yang lebih sederhana dan dapat ditafsirkan, seperti model linier, yang mendekati perilaku model asli secara lokal. Dengan menganalisis bobot dari model yang dapat dijelaskan, mereka dapat mengidentifikasi fitur-fitur yang paling berpengaruh dalam prediksi model asli. Penulis mendemonstrasikan keefektifan LIME di berbagai lingkup, termasuk *text classification* dan *image recognition*, memperlihatkan kemampuannya untuk memberikan wawasan kepada pengguna atau peneliti tentang alasan di balik keputusan model yang kompleks.

Selanjutnya, Lundberg dan Lee (2017) memperkenalkan SHAP (SHapley Additive exPlanations), sebuah *framework* untuk menginterpretasikan prediksi model. Penelitian mereka bertujuan untuk mengatasi kurangnya konsistensi dan landasan teori dalam fitur attribution method yang ada untuk menjelaskan model *machine learning*. SHAP didasarkan pada teori *game* kooperatif, yaitu setiap fitur dalam sebuah model diperlakukan sebagai "pemain" yang berkontribusi pada prediksi akhir. Metode ini memberikan nilai Shapley untuk setiap fitur, mengukur kontribusinya terhadap prediksi dengan cara yang matematis.

Lundberg dan Lee mendemonstrasikan pemanfaatan SHAP di berbagai model *machine learning*, termasuk model berbasis pohon dan jaringan syaraf tiruan. Mereka menerapkan SHAP pada dataset di dunia nyata, seperti data perawatan kesehatan untuk memprediksi hasil akhir pasien dan menunjukkan bagaimana SHAP dapat mengidentifikasi fitur-fitur utama yang memengaruhi prediksi. Penulis menekankan bahwa SHAP memberikan penjelasan yang konsisten dan akurat secara lokal, menjadikannya alat yang ampuh untuk menginterpretasikan model yang kompleks. Penelitian ini menjadi dasar teori bagi banyak penelitian selanjutnya yang mengintegrasikan SHAP ke dalam aplikasi *deep learning*.

Ribeiro, dkk. (2018) kemudian berkontribusi lagi pada XAI dengan mengembangkan Anchors, sebuah metode XAI untuk menghasilkan penjelasan *model-agnostic* dengan akurasi tinggi. Penulis bertujuan memberikan penjelasan

yang memberikan kepastian tentang perilaku model *machine learning* di area tertentu dari input. Anchors didefinisikan sebagai aturan atau kondisi yang, jika terpenuhi, memastikan bahwa prediksi model tetap konsisten dengan tingkat *confidence* yang tinggi.

Ribeiro dkk. mendemonstrasikan keefektifan Anchors dalam berbagai aplikasi, termasuk *text classification* dan *image recognition*. Mereka menunjukkan bahwa Anchors dapat mengidentifikasi aturan sederhana dan mudah ditafsirkan yang menjelaskan perilaku model yang kompleks, sehingga memberikan wawasan kepada pengguna tentang bagaimana model tersebut mengambil keputusan. Sebagai contoh, dalam *text classification*, Anchors dapat mengidentifikasi kata atau frasa tertentu yang, ketika ada dalam dokumen, secara konsisten menghasilkan hasil klasifikasi tertentu. Temuan mereka menunjukkan potensi Anchors sebagai alat yang berharga untuk memahami dan melakukan *debugging* pada model *machine learning*.

Terakhir, Kibriya dkk. (2024) menyelidiki penerapan XAI dalam deteksi dan klasifikasi ujaran kebencian menggunakan model *hybrid* yang menggabungkan jaringan CNN dan jaringan Bi-LSTM. Penelitian mereka bertujuan untuk mengatasi tantangan dua arah dalam mendeteksi ujaran kebencian secara akurat sekaligus memberikan penjelasan yang transparan untuk prediksi model. Mengingat sifat sensitif dari deteksi ujaran kebencian, memahami bagaimana keputusan dibuat sangat penting untuk memastikan pertanggungjawaban dan kepercayaan di antara para pengguna

Kibriya dkk. menerapkan LIME dan SHAP untuk menganalisis output model mereka. Teknik-teknik XAI ini secara efektif menekankan kata-kata dan frasa tertentu yang memainkan peran penting dalam mengklasifikasikan *tweet* sebagai ujaran kebencian atau bukan ujaran kebencian. Dengan memvisualisasikan penilaian terhadap tiap kata, penulis dapat menunjukkan bagaimana model *hybrid* mereka mencapai kesimpulan sekaligus mendorong transparansi yang lebih besar dalam sistem deteksi ujaran kebencian otomatis.

Dalam beberapa tahun terakhir, penelitian mengenai XAI semakin berkembang dan telah banyak diterapkan dalam berbagai bidang, terutama pada

sektor-sektor yang bersifat kritis seperti kesehatan (Tjoa dan Guan, 2021), keuangan (Arrieta dkk., 2020), serta sistem keamanan siber (Samek dkk., 2017). Penerapan XAI di bidang kesehatan telah memungkinkan dokter untuk memahami alasan di balik prediksi diagnosis model AI, sementara dalam bidang keuangan, XAI membantu dalam analisis risiko kredit dan keputusan investasi berbasis AI. Keberhasilan XAI dalam meningkatkan transparansi dan kepercayaan di bidang-bidang ini menunjukkan potensi penerapannya dalam tugas deteksi ujaran kebencian, dengan keputusan model AI dapat berdampak pada kebijakan media sosial dan regulasi hukum.

Selain itu, studi yang dilakukan oleh (Arras dkk., 2017) menunjukkan bagaimana XAI dapat membantu dalam klasifikasi teks dengan menyoroti kata-kata yang memiliki kontribusi signifikan terhadap keputusan model. Dengan demikian, penggunaan metode interpretabilitas seperti LIME dan SHAP tidak hanya meningkatkan transparansi model, tetapi juga membantu dalam mendeteksi potensi bias dalam sistem klasifikasi ujaran kebencian. Dengan meningkatnya penerapan XAI di berbagai sektor, integrasi XAI dalam deteksi ujaran kebencian menjadi suatu langkah yang logis dan mendukung pengembangan AI yang lebih dapat dipercaya.

Tabel 2.2 merangkum beberapa metode utama dalam XAI yang telah diperkenalkan dalam berbagai penelitian. Ribeiro dkk. memperkenalkan LIME, sebuah metode yang membangun model sederhana untuk mendekati perilaku model kompleks secara lokal, memberikan wawasan terhadap keputusan model. Lundberg dan Lee mengembangkan SHAP, yang menggunakan nilai Shapley untuk mengukur kontribusi setiap fitur dalam model, memberikan penjelasan yang lebih akurat dan konsisten. Terakhir, Ribeiro dkk. mengembangkan Anchors, yang menggunakan aturan berbasis *high confidence* untuk memberikan interpretasi model yang lebih mudah dipahami. Dengan membandingkan metode-metode ini, terlihat bahwa masing-masing teknik memiliki pendekatan yang berbeda dalam menjelaskan model *machine learning*, dengan LIME berfokus pada kesederhanaan lokal, SHAP memberikan penjelasan yang lebih teoritis berdasarkan nilai Shapley, dan Anchors menyajikan interpretasi berbasis aturan yang lebih intuitif.

Tabel 2.2 Perbandingan studi penelitian XAI

Penulis	**Metode yang Diperkenalkan**	**Pendekatan yang digunakan**	**Kelebihan Metode**
(Ribeiro dkk., 2016)	LIME	Membuat model sederhana yang mendekati perilaku model asli secara lokal.	Memberikan wawasan lokal terhadap keputusan model kompleks.
(Lundberg dan Lee, 2017)	SHAP	Menggunakan nilai Shapley untuk mengukur kontribusi tiap fitur.	Memberikan penjelasan yagn konsisten dan akurat secara lokal.
(Ribeiro dkk., 2018)	Anchors	Menggunakan aturan sederhana dengan *confidence* tinggi untuk menjelaskan prediksi model.	Memungkinkan interpretasi berbasis aturan yang lebih mudah dipahami.

BAB III

LANDASAN TEORI

3.1 Ujaran Kebencian

Ujaran kebencian dapat didefinisikan sebagai ujaran atau ungkapan yang dimaksudkan untuk menyerang, mengancam, memicu kekerasan, atau merendahkan individu atau kelompok berdasarkan karakteristik seperti ras, agama, asal etnis, orientasi seksual, disabilitas, atau jenis kelamin (Brown, 2017). Unsur utama dari ujaran kebencian terletak pada sifatnya yang diskriminatif dan menghasut, yang berpotensi menimbulkan permusuhan, prasangka, atau bahkan kekerasan terhadap kelompok yang menjadi sasaran (Waldron, 2012). Meskipun sering dikaitkan dengan permusuhan secara eksplisit, ujaran kebencian juga bisa muncul secara implisit, sehingga sulit untuk dikenali dan dikendalikan secara konsisten di berbagai konteks sosial dan budaya (Brown, 2017).

Dalam sejarahnya, ujaran kebencian telah dipelajari secara luas dalam psikologi sosial dan hukum, yang menekankan dampaknya terhadap keharmonisan sosial, kehormatan individu, dan diskusi demokratis (Waldron, 2012). Kompleksitas dalam mendefinisikan ujaran kebencian muncul dari sensitivitas dan standar budaya yang berbeda di berbagai masyarakat, yang mengarah pada perdebatan yang sedang berlangsung tentang menyeimbangkan kebebasan berekspresi dengan perlindungan dari ujaran yang berbahaya (Hare dan Weinstein, 2009).

Di Indonesia, regulasi mengenai ujaran kebencian telah diatur dalam Surat Edaran Kapolri No. SE/6/X/2015 yang mengklasifikasikan ujaran kebencian sebagai tindak pidana yang mencakup penghinaan, perbuatan tidak menyenangkan, provokasi, penghasutan, penyebaran berita bohong, serta tindakan lain yang dapat memicu diskriminasi, kekerasan, atau konflik sosial. SE ini juga mengatur berbagai media yang dapat digunakan untuk menyebarkan ujaran kebencian, seperti media sosial, kampanye, spanduk, ceramah keagamaan, dan media cetak. Selain itu, aparat penegak hukum diberikan kewenangan untuk melakukan tindakan preventif dan represif guna menekan penyebaran ujaran kebencian yang dapat mengganggu

stabilitas sosial dan keamanan nasional (Kepolisian Negara Republik Indonesia, 2015). Dengan adanya regulasi ini, sistem deteksi ujaran kebencian otomatis dapat membantu penegakan hukum dalam mengidentifikasi potensi ujaran kebencian secara lebih efisien dan akurat.

3.2 Pemrosesan Bahasa Alami

Pemrosesan Bahasa Alami atau *natural language processing* (NLP) adalah cabang dari *artificial intelligence* (AI) yang berfokus pada kemampuan komputer untuk memproses, menganalisis, dan menghasilkan bahasa manusia (Jurafsky dan Martin, 2008). NLP menggabungkan komputasi linguistik, model *rule-based*, dan *machine learning* untuk menangani berbagai tugas yang berhubungan dengan bahasa seperti klasifikasi teks, analisis sentimen, dan mesin penerjemah (Manning dan Schütze, 1999). Karena bahasa manusia pada dasarnya kompleks, NLP melibatkan beberapa tahap pemrosesan, termasuk pra-pemrosesan teks (tokenisasi, penghilangan *stopword*, dan *stemming*), analisis sintaksis (*part-of-speech tagging* dan *parsing*), dan analisis semantik (*named entity recognition* dan *word sense disambiguation*) untuk mengekstrak struktur yang bermakna dari teks yang tidak terstruktur (Jurafsky dan Martin, 2008). Kemajuan terbaru dalam *deep learning*, khususnya arsitektur berbasis *transformer* seperti BERT, telah secara signifikan meningkatkan kemampuan NLP dengan meningkatkan pemahaman bahasa kontekstual (Vaswani dkk., 2017).

Aplikasi NLP dapat dikategorikan secara luas ke dalam *natural language understanding* (NLU) dan *natural language generation* (NLG) (Manning dan Schütze, 1999). NLU berfokus pada penafsiran dan pengambilan informasi dari teks, seperti dalam analisis sentimen, menjawab pertanyaan, dan peringkasan teks. Sementara itu, NLG melibatkan produksi teks yang tepat dan sesuai dengan konteks, yang biasa digunakan dalam *chatbot*, mesin penerjemah, dan *text summarization* (Reiter dan Dale, 2000). Kemajuan yang terus menerus dalam NLP, terutama dengan model berbasis *transformer*, telah secara signifikan meningkatkan kinerja NLU dan NLG, membuat sistem AI berbasis bahasa menjadi lebih akurat dan dapat memahami konteks (Vaswani dkk., 2017).

Tabel 3.1 membandingkan NLU dan NLG berdasarkan beberapa aspek utama. NLU berfokus pada pemrosesan informasi dari teks, termasuk tugas seperti analisis sentimen, menjawab pertanyaan, dan peringkasan teks. Sebaliknya, NLG bertujuan untuk menghasilkan teks yang jelas dan sesuai konteks, yang digunakan dalam *chatbot*, *machine translation*, dan *text summarization*. Dari segi pendekatan, NLU menggunakan teknik berbasis linguistik dan *deep learning*, sedangkan NLG banyak memanfaatkan *transformer-based models* seperti BERT dan GPT. Dengan memahami perbedaan ini, dapat disimpulkan bahwa NLU dan NLG saling melengkapi dalam sistem NLP yang lebih kompleks.

Tabel 3.1 Perbandingan NLU dengan NLG

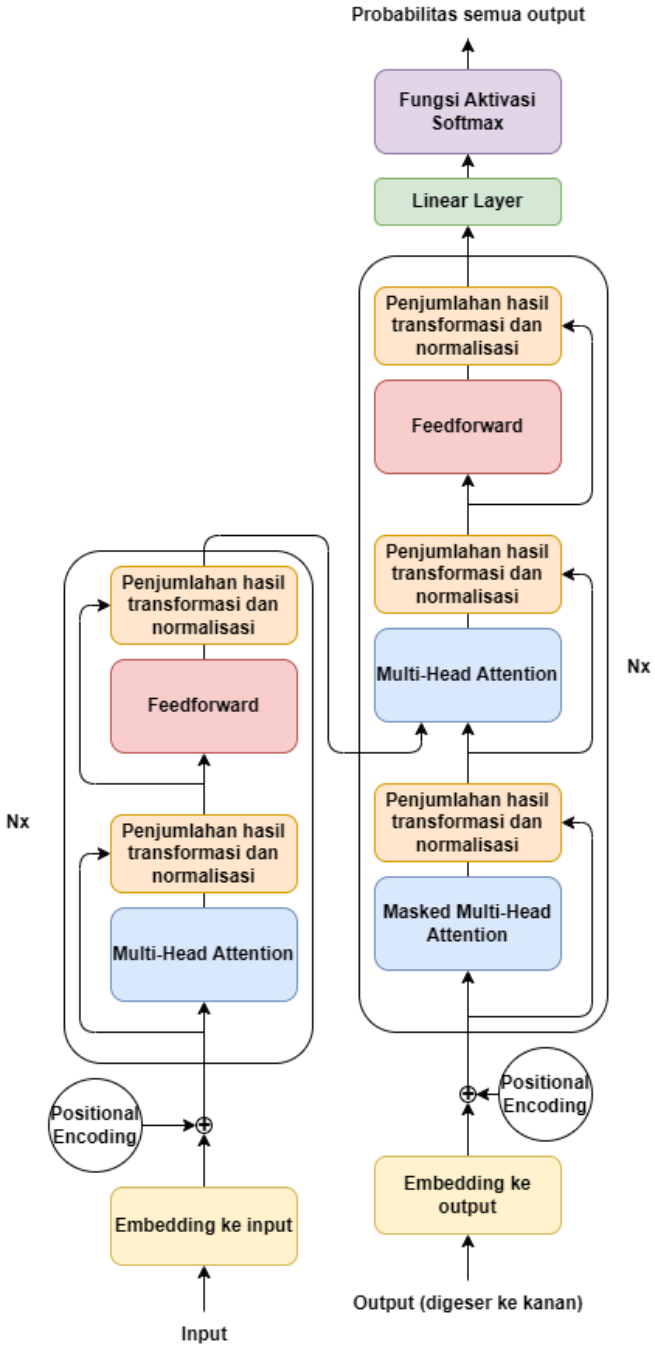
Aspek	NLU	NLG
Definisi	Memproses dan mengekstrak informasi dari teks.	Menghasilkan teks yang jelas dan sesuai konteks.
Fokus Utama	Memahami dan menginterpretasikan bahasa alami.	Menyusun dan menghasilkan teks dalam bahasa alami.
Tugas Utama	● *Sentiment analysis* ● *Question answering* ● *Text summarization*	● *Chatbot response generation* ● *Machine translation* ● *Text summarization*
Teknologi yang Digunakan	Pendekatan berbasis linguistik dan *deep learning.*	*Transformer-based models* seperti BERT dan GPT.

3.3 Transformer dan BERT

Transformer adalah arsitektur *deep learning* yang diperkenalkan oleh Vaswani dkk. (2017) dan telah merevolusi NLP dengan menggantikan model berbasis *recurrent* dan *convolutional* dengan mekanisme yang sepenuhnya menggunakan *self-attention*. Berbeda dengan model sebelumnya seperti Recurrent Neural Networks (RNNs) dan LSTM, *transformer* dapat memproses keseluruhan urutan teks secara paralel, sehingga lebih efisien dalam menangani *long-range dependencies* pada teks (Vaswani dkk., 2017). Keunggulan ini dicapai melalui dua komponen utama, yaitu *self-attention* dan *positional encoding*, yang memungkinkan model untuk secara dinamis menentukan bagian mana dari sebuah

kalimat yang paling relevan dalam memahami konteksnya. Karena kemampuannya yang unggul dalam menangani teks, arsitektur *transformer* telah menjadi dasar dari berbagai model NLP modern seperti GPT, T5, dan BERT.

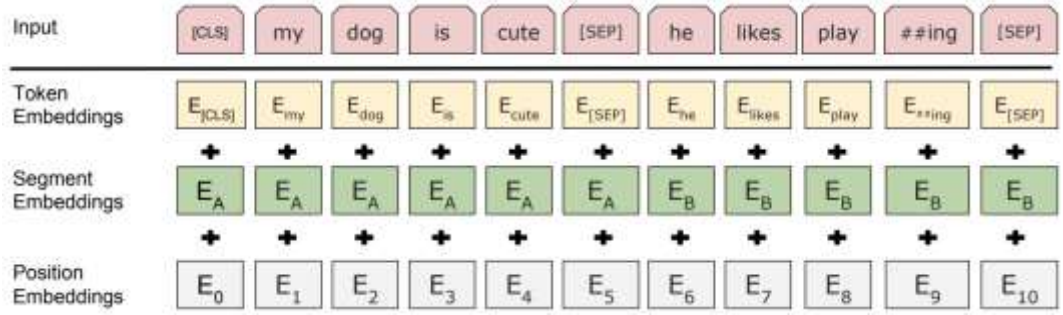
Gambar 3.1 menggambarkan arsitektur *transformer*, yang terdiri dari dua bagian utama, yaitu *encoder* dan *decoder*. *Encoder* bertugas untuk memproses input dengan serangkaian blok *multi-head attention* dan *feedforward neural networks*, yang masing-masing diikuti oleh proses normalisasi dan *residual connection*. Sementara itu, *decoder* menggunakan struktur yang serupa dengan tambahan *masked multi-head attention*, yang mencegah model melihat token di depan selama proses pelatihan. Setelah melalui beberapa *layer*, hasil akhirnya diproses melalui *linear layer* dan fungsi aktivasi *softmax* untuk menghasilkan output akhir berupa probabilitas kata yang diprediksi. Dengan struktur ini, *transformer* mampu memahami hubungan kompleks antar kata dalam sebuah teks dengan lebih efisien dibandingkan model sebelumnya.



Gambar 3.1 Arsitektur *transformer* (Vaswani dkk., 2017)

Pada penerapan *transformer* untuk NLP, model menerima input teks dalam bentuk token yang direpresentasikan sebagai vektor *embedding*. Selain token *embeddings*, model juga menggunakan *segment embeddings* untuk membedakan antara dua kalimat berbeda dalam input, serta *position embeddings* yang mempertahankan urutan kata dalam sebuah kalimat. Proses ini ditunjukkan pada

Gambar 3.2, yang memperlihatkan representasi input dalam BERT. Setiap kata dalam input direpresentasikan sebagai gabungan dari *token embedding*, *segment embedding*, dan *position embedding* yang kemudian digunakan dalam proses *self-attention* untuk memahami hubungan antar kata dalam teks.



Gambar 3.2 Representasi input dalam BERT (Devlin et al., 2019)

BERT, yang diperkenalkan oleh (Devlin dkk., 2019), merupakan salah satu model berbasis *transformer* yang memiliki dampak signifikan dalam pemrosesan teks. Berbeda dengan teknik *embedding* kata konvensional yang memberikan vektor tetap untuk setiap kata, BERT mampu menangkap makna kontekstual secara *bidirectional* melalui proses *pre-training* pada korpus teks berskala besar menggunakan dua teknik utama, yaitu Masked Language Modeling (MLM) dan Next Sentence Prediction (NSP).

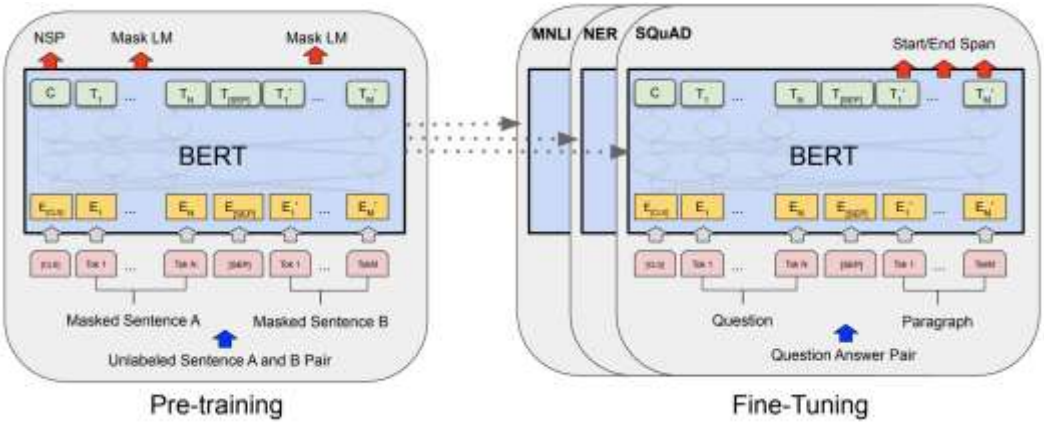
Pada *Masked Language Modeling* (MLM), sejumlah token dalam teks input digantikan dengan token spesial $[MASK]$, dan model bertugas untuk memprediksi kata asli yang tersembunyi berdasarkan konteks yang tersedia. Untuk setiap token yang di-*masking*, model menghasilkan distribusi probabilitas seperti pada Persamaan (3.1) yang merepresentasikan kemungkinan setiap kata dalam kosakata sebagai pengganti token yang di-*masking*.

$$P(x_t|X_{\setminus t}), \tag{3.1}$$

Model dilatih untuk meminimalkan perbedaan antara token asli x_t dan prediksi model berdasarkan fungsi *loss* berbasis *cross-entropy*, sehingga semakin kecil perbedaan antara prediksi dan kata asli, semakin baik performa model dalam memahami konteks suatu kata dalam kalimat.

Selain MLM, BERT juga menggunakan *Next Sentence Prediction* (NSP) untuk memahami hubungan antar-kalimat. Dalam pekerjaan ini, model diberikan dua kalimat, A dan B, dan harus menentukan apakah B merupakan kelanjutan dari A dalam teks aslinya. Model mempelajari distribusi probabilitas $P(S|A,B)$ untuk menentukan apakah kalimat B memang merupakan lanjutan dari A, serta $P(N|A,B)$ untuk mendeteksi jika B tidak berhubungan dengan A. Model kemudian mengoptimalkan bobotnya untuk memaksimalkan prediksi yang benar, menggunakan pendekatan klasifikasi biner dengan *loss* berbasis *cross-entropy* antara label *ground truth* y_{AB} dan probabilitas prediksi model. Dengan adanya NSP, BERT tidak hanya memahami makna kata dalam kalimat, tetapi juga relasi antar-kalimat, yang membedakannya dari model *embedding* sebelumnya seperti Word2Vec dan GloVe.

Gambar 3.3 menggambarkan secara rinci bagaimana proses pre-training dan fine-tuning dilakukan dalam model BERT. Pada tahap *pre-training*, model dilatih dengan menggunakan Masked Language Modeling (MLM) dan Next Sentence Prediction (NSP) pada data tanpa label. Model yang telah dilatih kemudian dapat digunakan untuk berbagai tugas NLP melalui proses *fine-tuning*, dengan semua *parameter* model yang dapat disesuaikan menggunakan dataset spesifik dari tugas yang ingin diselesaikan, seperti *text classification* atau *entity extraction*.



Gambar 3.3 *Pre-training* dan *fine-tuning* BERT (Devlin dkk., 2019)

3.3.1 WordPiece Tokenization

WordPiece Tokenization adalah teknik tokenisasi berbasis sub-kata yang digunakan dalam model *transformer* seperti BERT untuk menangani kata-kata *out-of-vocabulary* (OOV). Teknik ini bekerja dengan membagi kata menjadi unit sub-kata yang lebih kecil dan kemudian menggabungkannya berdasarkan aturan probabilistik yang dihasilkan dari data *training* (Schuster dan Nakajima, 2012). Proses segmentasi dalam *WordPiece* biasanya mengadopsi pendekatan *Maximum Matching* (*MaxMatch*), yang memilih prefiks terpanjang dari sebuah kata yang sesuai dengan kosakata yang telah dilatih. Namun, pendekatan ini memiliki keterbatasan dalam efisiensi pencarian token, terutama saat menangani teks dalam skala besar. Untuk mengatasi hal tersebut, penelitian terbaru telah mengusulkan berbagai optimasi, salah satunya adalah *LinMaxMatch Tokenization*, yang meningkatkan kecepatan segmentasi dengan menggunakan struktur *tree* dan *failure transitions* untuk menghindari *backtracking* yang tidak perlu (Song dkk., 2021).

Dalam *library* Hugging Face Transformers, *WordPiece Tokenization* diimplementasikan untuk memastikan kompatibilitas dengan berbagai model berbasis BERT, termasuk HateBERT. Hugging Face Tokenizer menyediakan dua varian utama: *Python-based Tokenizer* dan *Fast Tokenizer* berbasis Rust, dengan versi *Fast Tokenizer* yang menawarkan kecepatan pemrosesan lebih tinggi. Proses tokenisasi terdiri dari beberapa tahapan, yaitu normalisasi teks, tokenisasi berbasis *WordPiece*, serta konversi token ke ID numerik yang dapat langsung digunakan oleh model. *WordPiece Tokenization* dalam Hugging Face bekerja dengan mencari sub-kata yang paling sesuai dalam kosakata model, menggunakan pendekatan *Maximum Matching* yang telah dioptimalkan untuk kecepatan dan akurasi. Dalam implementasi mereka, struktur *tree* digunakan untuk mempercepat pencarian token, menghindari duplikasi proses tokenisasi, dan meningkatkan efisiensi pemrosesan teks dalam skala besar (Hugging Face, 2023).

3.3.2 Mekanisme Self-Attention

Mekanisme *self-attention* adalah inti dari arsitektur *transformer*, yang memungkinkan model untuk mempertimbangkan hubungan antara elemen-elemen dalam sebuah urutan data secara efektif. Dalam konteks pemrosesan bahasa alami,

mekanisme ini memungkinkan model untuk memahami konteks sebuah kata berdasarkan kata-kata lain dalam kalimat tersebut. Setiap kata dalam input diubah menjadi tiga vektor: *Query* (Q), *Key* (K), dan *Value* (V). Proses *self-attention* menghitung skor kesamaan antara vektor *Query* dan *Key* untuk menentukan seberapa banyak perhatian yang harus diberikan pada setiap kata saat merepresentasikan sebuah kata tertentu. Skor ini kemudian dinormalisasi menggunakan fungsi *SoftMax*, dan hasilnya digunakan untuk menghitung rata-rata tertimbang dari vektor *Value*, menghasilkan representasi baru untuk setiap kata yang telah mempertimbangkan konteksnya dalam kalimat (Vaswani dkk., 2017).

Persamaan (3.2) memperlihatkan mekanisme *self-attention* secara matematis (Vaswani dkk., 2017):

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.2)$$

Di sini, Q, K, dan V adalah matriks *Query*, *Key*, dan *Value*; d_k adalah dimensi dari vektor *Key*. Pembagian dengan $\sqrt{d_k}$ digunakan untuk menstabilkan nilai *dot product* agar tidak terlalu besar, yang dapat menyebabkan gradien yang kecil saat menggunakan fungsi *SoftMax* (Vaswani dkk., 2017). Dalam model seperti BERT, mekanisme *self-attention* memungkinkan model untuk memahami makna kata secara kontekstual dengan mempertimbangkan hubungan antara kata-kata dalam sebuah kalimat, sehingga meningkatkan akurasi dalam berbagai tugas pemrosesan bahasa alami (Devlin dkk., 2019).

3.4 HateBERT

HateBERT adalah model bahasa berbasis BERT yang telah dilatih ulang khusus untuk mendeteksi bahasa kasar dan ujaran kebencian dalam bahasa Inggris. Model ini dilatih menggunakan RAL-E, sebuah dataset yang terdiri dari komentar-komentar di platform Reddit dalam Bahasa Inggris yang berasal dari komunitas yang dilarang karena dianggap ofensif, kasar, atau penuh kebencian. Tujuan utama dari pelatihan ulang ini adalah untuk meningkatkan kemampuan model dalam mengenali dan mengklasifikasikan bahasa yang bersifat abusif atau penuh kebencian di platform media sosial (Caselli dkk., 2021).

Secara arsitektur, HateBERT mempertahankan struktur dasar dari BERT,

yang terdiri dari lapisan-lapisan *transformer* dengan mekanisme *self-attention*. Namun, perbedaan utama terletak pada data pelatihan yang digunakan; dengan melatih ulang BERT pada data ujaran kebencian, HateBERT dapat memahami konteks yang lebih baik dalam mendeteksi jenis bahasa tersebut. Meskipun tidak ada perubahan signifikan dalam rumus atau mekanisme internal dibandingkan dengan BERT asli, pelatihan ulang pada dataset spesifik ini memungkinkan HateBERT untuk lebih efektif dalam tugas deteksi ujaran kebencian (Caselli dkk., 2021).

3.5 Transfer Learning

Transfer learning adalah teknik dalam *machine learning* yang bekerja dengan cara pengetahuan yang diperoleh dari satu tugas digunakan kembali untuk meningkatkan kinerja pada tugas lain yang terkait. Misalnya, model yang telah dilatih untuk mengenali objek umum dapat digunakan sebagai dasar untuk mengenali objek spesifik dengan jumlah data yang lebih sedikit. Pendekatan ini berbeda dari metode tradisional yang melatih model dari awal untuk setiap tugas spesifik, yang memerlukan data dan sumber daya komputasi yang besar. *Transfer learning* memungkinkan penggunaan kembali model yang telah dilatih sebelumnya, sehingga menghemat waktu dan sumber daya, serta meningkatkan kinerja terutama ketika data pada tugas baru terbatas (Ruder dkk., 2019).

Dalam konteks NLP, *transfer learning* biasanya melibatkan dua tahap utama: *pre-training* dan *fine-tuning*. Pada tahap *pre-training*, model dilatih pada korpus teks besar untuk mempelajari representasi bahasa umum. Setelah itu, pada tahap *fine-tuning*, model yang telah dilatih ini disesuaikan dengan tugas spesifik, seperti analisis sentimen atau deteksi ujaran kebencian, menggunakan dataset yang lebih kecil namun relevan. Pendekatan ini telah terbukti efektif dalam meningkatkan kinerja model NLP pada berbagai tugas, termasuk klasifikasi teks dan *named-entity recognition* (NER) (Ruder dkk., 2019).

Dalam penelitian ini, proses *transfer learning* dilakukan pada model HateBERT yang telah melalui tahap *pre-training* menggunakan data ujaran kebencian dari Reddit berjudul RAL-E (Caselli dkk., 2021). Tahap yang digunakan dalam penelitian ini adalah *fine-tuning*, yaitu penyesuaian *parameter* model

HateBERT dengan dataset baru, yaitu OffensEval 2019, yang belum pernah digunakan dalam pelatihan awal model. Proses ini memungkinkan model untuk men-*transfer* pengetahuan yang telah diperoleh dari domain awal ke domain target yang baru, sesuai definisi *transfer learning* (Pan dan Yang, 2010).

Transfer learning merupakan metode yang berbeda dengan penggunaan *pre-trained model* secara langsung. Penggunaan *pre-trained model* tanpa melakukan *fine-tuning* berarti model digunakan apa adanya, hanya untuk inferensi, tanpa adaptasi terhadap data baru. Sebaliknya, dalam *transfer learning*, model *pre-trained* digunakan sebagai titik awal, kemudian dilatih ulang (*fine-tuning*) pada dataset target agar model dapat mengakomodasi karakteristik domain baru (Zhuang dkk., 2020). Oleh karena itu, proses yang dilakukan dalam penelitian ini, yaitu *fine-tuning* HateBERT pada dataset OffensEval 2019, merupakan bentuk dari *transfer learning*.

Secara matematis, *transfer learning* melibatkan perpindahan pengetahuan dari satu domain ke domain lainnya. Jika D_S dan D_T masing-masing merepresentasikan domain sumber dan domain target, maka *transfer learning* terjadi ketika ada perbedaan antara domain atau tugas yang dikerjakan, yakni $D_S \neq D_T$ atau $T_S \neq T_T$. Fungsi prediktif pada domain target, $fr(\cdot)$, dioptimalkan dengan menggunakan informasi yang diperoleh dari domain sumber, sehingga model dapat mempelajari pola dari tugas awal dan menerapkannya pada tugas baru yang memiliki karakteristik serupa (Ruder dkk., 2019).

3.6 Pra-Pemrosesan Teks

Pra-pemrosesan teks adalah tahap awal dalam NLP yang bertujuan untuk membersihkan dan menyiapkan data teks mentah agar siap untuk analisis lebih lanjut atau masuk ke fase *training*. Langkah-langkah umum dalam pra-pemrosesan teks meliputi tokenisasi, normalisasi (seperti pengubahan huruf besar menjadi huruf kecil), dan *text* cleaning atau pembersihan *noise* pada teks yang dapat mengganggu proses *training* model. Proses ini bertujuan untuk mengurangi kompleksitas data dan meningkatkan kinerja model NLP. Penelitian menunjukkan bahwa keputusan pra-pemrosesan yang sederhana, seperti tokenisasi dan *text cleaning*, dapat memiliki dampak signifikan terhadap kinerja model klasifikasi teks dan analisis

sentimen (Camacho-Collados dan Pilehvar, 2018). Selain itu, pendekatan baru dalam evaluasi pra-pemrosesan NLP telah diusulkan untuk memastikan evaluasi yang akurat (Wiącek dkk., 2024).

3.6.1 Padding dan Truncation

Padding dan *Truncation* adalah teknik penting dalam pra-pemrosesan teks yang digunakan untuk memastikan keseragaman panjang input dalam model *machine learning*, terutama saat menangani data berurutan seperti teks. *Padding* melibatkan penambahan teks kosong atau nilai tertentu ke input yang lebih pendek sehingga mencapai panjang yang diinginkan. Misalnya, jika sebuah model memerlukan input dengan panjang tetap n dan sebuah urutan memiliki panjang m (dengan $m < n$), maka $n - m$ elemen *padding* ditambahkan ke urutan tersebut (Feldman dan Sanger, 2006). Sebaliknya, *truncation* adalah proses memotong input yang lebih panjang sehingga sesuai dengan panjang maksimum yang ditentukan. Jika sebuah urutan memiliki panjang m (dengan $m > n$), maka hanya n elemen pertama atau terakhir yang dipertahankan, sementara sisanya dihapus. Tujuan utama dari kedua teknik ini adalah untuk memastikan bahwa semua input memiliki panjang yang konsisten, memungkinkan pemrosesan *batch* yang efisien dan kompatibilitas dengan arsitektur model yang memerlukan panjang input yang konsisten (Mutasodirin dan Prasojo, 2021).

Dalam konteks pemodelan teks, penerapan *padding* dan *truncation* memiliki implikasi signifikan terhadap kinerja model. Penelitian menunjukkan bahwa penggunaan *padding* cenderung menghasilkan akurasi klasifikasi yang lebih tinggi dibandingkan dengan *truncation*, terutama karena *padding* mempertahankan informasi dalam urutan asli tanpa kehilangan data penting yang mungkin terjadi selama proses *truncation*. Selain itu, *padding* memungkinkan model untuk memanfaatkan konteks penuh dari urutan, sementara *truncation* dapat menyebabkan hilangnya informasi kritis yang berada di luar batas panjang yang ditentukan. Oleh karena itu, pemilihan antara *padding* dan *truncation* harus dilakukan dengan mempertimbangkan karakteristik data dan kebutuhan spesifik dari tugas pemodelan yang sedang dilakukan (Li, 2024).

3.6.2 Encoding Label

Encoding Label adalah proses mengubah label kategorikal menjadi representasi berupa angka agar dapat diproses oleh algoritma *machine learning*. Salah satu metode yang paling umum digunakan adalah *Label Encoding*, setiap kategori unik diberi nilai integer. Misalnya, untuk tiga kategori "Merah", "Hijau", dan "Biru", kita dapat menetapkan "Merah" = 0, "Hijau" = 1, dan "Biru" = 2. Pendekatan ini memiliki kelemahan karena algoritma dapat menganggap ada urutan antar kategori yang sebenarnya tidak memiliki urutan, yang dapat menyebabkan kesalahan interpretasi dalam proses pembelajaran model *machine learning* (Johannemann dkk., 2021). Oleh karena itu, untuk menghindari asumsi hubungan ordinal, metode *One-Hot Encoding* lebih sering digunakan.

Dalam *One-Hot Encoding*, setiap kategori direpresentasikan sebagai vektor biner dengan panjang yang sama dengan jumlah kategori yang tersedia. Misalnya, untuk tiga kategori "Merah", "Hijau", dan "Biru", representasi *one-hot* masing-masing adalah $[1, 0, 0]$, $[0, 1, 0]$, dan $[0, 0, 1]$. Pendekatan ini lebih cocok untuk algoritma *machine learning* yang mempertimbangkan nilai numerik sebagai ukuran kesamaan antar kategori. Namun, keunggulan utama dari *One-Hot Encoding* adalah kemampuannya untuk mengubah kategori menjadi angka tanpa memberikan makna tambahan yang bisa membuat model melakukan kesalahan, meskipun metode ini membutuhkan lebih banyak daya komputasi (Rodríguez dkk., 2018). Dalam konteks model BERT, *label encoding* biasanya dilakukan dengan mengubah label ke dalam indeks numerik sebelum digunakan dalam model klasifikasi. Selama pelatihan, model akan memprediksi probabilitas untuk setiap kelas, dan fungsi *loss* seperti *Cross-Entropy Loss* akan digunakan untuk mengoptimalkan prediksi berdasarkan label yang telah di-*encoding* (Wang dkk., 2021).

3.7 Metrik Evaluasi

Dalam konteks *transfer learning* dengan model HateBERT untuk tugas deteksi ujaran kebencian, penting untuk memilih metrik evaluasi yang mampu menangkap performa model secara adil dan menyeluruh. Deteksi ujaran kebencian merupakan masalah klasifikasi yang rentan terhadap ketidakseimbangan kelas, dengan jumlah contoh ujaran kebencian biasanya jauh

lebih sedikit dibanding contoh yang bukan ujaran kebencian. Oleh karena itu, akurasi saja tidak cukup merepresentasikan kemampuan model secara utuh. Metrik seperti *precision*, *recall*, *macro F1-score*, dan *positive* (POS) *F1-score* digunakan untuk menilai secara spesifik seberapa baik model dalam mendeteksi dan membedakan ujaran kebencian dari teks netral, serta mengurangi konsekuensi kesalahan klasifikasi.

3.7.1 Precision

Precision penting dalam konteks deteksi ujaran kebencian karena bertujuan mengurangi kesalahan tipe I (*false positives*), yaitu ketika teks non-kebencian secara keliru diklasifikasikan sebagai ujaran kebencian. Dalam aplikasi nyata, hal ini dapat menyebabkan laporan atau penyensoran yang tidak adil terhadap pengguna yang tidak melakukan pelanggaran. Persamaan (3.3) merupakan persamaan untuk menghitung nilai *precision* (Powers, 2011):

$$Precision = \frac{TP}{TP + FP} \tag{3.3}$$

Pada persamaan (3.3), TP adalah jumlah *true positives* atau jumlah prediksi benar untuk kelas kebencian dan FP adalah jumlah *false positives* atau teks non-kebencian yang salah diklasifikasikan sebagai ujaran kebencian. *precision* memberikan wawasan tentang seberapa akurat model dalam mengidentifikasi kelas positif tanpa memasukkan terlalu banyak kesalahan. Nilai *precision* yang tinggi menandakan bahwa model dapat dipercaya ketika memberikan label kebencian, yang penting dalam sistem moderasi otomatis.

3.7.2 Recall

Recall lebih menekankan pada kemampuan model untuk menangkap semua contoh ujaran kebencian yang ada dan menghindari kesalahan tipe II atau *false negatives*. Dalam banyak kasus, terutama pada platform media sosial, kegagalan mendeteksi ujaran kebencian dapat berdampak serius terhadap keamanan dan kenyamanan pengguna. Persamaan (3.4) merupakan persamaan untuk menghitung nilai *recall* (Powers, 2011):

$$Recall = \frac{TP}{TP + FN} \tag{3.4}$$

Pada persamaan (3.4), TP adalah jumlah *true positives* atau ujaran kebencian yang berhasil dideteksi oleh model dan FN adalah jumlah *false negatives* atau jumlah teks ujaran kebencian yang tidak dikenali oleh model. Metrik ini memberikan wawasan tentang seberapa lengkap model dalam menangkap semua konten berbahaya, yang penting dalam sistem yang harus mencegah penyebaran ujaran kebencian secara efektif.

3.7.3 Macro F1-Score

Macro F1-score cocok untuk deteksi ujaran kebencian yang cenderung memiliki distribusi kelas yang tidak seimbang. Metrik ini memperlakukan semua kelas secara setara, menghitung *precision* dan *recall* untuk masing-masing kelas secara terpisah, lalu dirata-ratakan, sehingga performa pada kelas minoritas, dalam hal ini ujaran benecian, tidak tertutupi oleh performa tinggi di kelas mayoritas. Persamaan (3.5) merupakan persamaan untuk menghitung nilai *macro F1-score* (Powers, 2011):

$$Macro\ F1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times Precision_i \times Recall_i}{Precision_i + Recall_i} \tag{3.5}$$

Pada persamaan (3.5), N adalah jumlah kelas, dan $Precision_i$ serta $Recall_i$ adalah *precision* dan *recall* untuk kelas ke-i. Metrik ini memastikan bahwa kinerja model dievaluasi secara adil di semua kelas, tanpa terpengaruh oleh distribusi kelas yang tidak seimbang.

3.7.4 Positive (POS) F1-Score

POS F1-score difokuskan hanya pada kelas positif, yaitu kelas ujaran kebencian. Ini berguna ketika perhatian utama terdapat pada kemampuan model dalam mendeteksi kategori yang berbahaya. Metrik ini digunakan untuk menilai seberapa baik model dalam mengenali ujaran kebencian secara spesifik tanpa mempertimbangkan kelas netral. Persamaan (3.6) merupakan persamaan untuk menghitung nilai *POS F1-score* (Powers, 2011):

$$POS\ F1 = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times Precision_{POS} \times Recall_{POS}}{Precision_{POS} + Recall_{POS}} \tag{3.6}$$

Pada persamaan (3.6) $Precision_{POS}$ dan $Recall_{POS}$ adalah *precision* dan *recall* khusus untuk kelas positif atau ujaran kebencian. Metrik ini penting untuk memastikan bahwa model tidak hanya mengenali instans positif dengan benar tetapi juga mengklasifikasikannya secara akurat. Nilai *POS F1-score* yang tinggi menunjukkan bahwa model mampu secara konsisten dan benar mengidentifikasi ujaran kebencian, menjadikannya metrik penting dalam konteks deteksi teks berbahaya secara spesifik.

3.8 Local Interpretable Model-agnostic Explanations (LIME)

LIME adalah teknik yang dirancang untuk menjelaskan prediksi dari model *machine learning* yang kompleks, sering disebut sebagai "*black-box models*". LIME berfungsi dengan membuat model sederhana yang dapat menjelaskan alasan di balik suatu keputusan model untuk sebuah input tertentu, memungkinkan pengguna atau peneliti untuk memahami alasan di balik keputusan model tersebut. Pendekatan ini pertama kali diperkenalkan oleh Ribeiro dkk. dalam makalah berjudul "Why Should I Trust You?": Explaining the Predictions of Any Classifier (Ribeiro dkk., 2016).

Secara operasional, LIME bekerja dengan membuat variasi dari data asli melalui perubahan kecil pada fitur input dan mengamati dampaknya terhadap output model. Dengan melakukan ini, LIME membangun model yang dapat dilihat penjelasannya, seperti *linear regression* atau *decision tree*, yang mendekati perilaku model kompleks dalam lingkup lokal. Model ini kemudian digunakan untuk mengidentifikasi kontribusi masing-masing fitur terhadap prediksi, memberikan *local interpretability* terhadap keputusan model (Ribeiro dkk., 2016).

Secara matematis, LIME mencari model interpretable g yang meminimalkan local loss function $L(f, g, \pi_x)$, yang mengukur sejauh mana $g$mendekati model yang lebih kompleks f dalam lingkup lokal π_x, sambil mempertahankan model dengan kompleksitas minimal. Persamaan (3.7) memperlihatkan pertanyaan formulasi tersebut (Molnar, 2022):

$$\text{explanation}(x) = \arg\min_{g \in G} L(f, g, \pi_x) + \Omega_{(g)} \tag{3.7}$$

Fungsi *local loss* $L(f, g, \pi_x)$ merepresentasikan *local fidelity* atau tingkat kesesuaian lokal, yang mengukur sejauh mana model yang dapat dijelaskan

$g$mereplikasi perilaku model asli f dalam wilayah sekitarnya π_x. *Regularization term* $\Omega_{(g)}$ digunakan untuk mengontrol kompleksitas model yang dapat dijelaskan, memastikan keseimbangan antara seberapa mudah penjelasannya untuk dipahami dan seberapa baik model sederhana ini meniru model yang lebih kompleks (Molnar, 2022).

BAB IV

METODE PENELITIAN

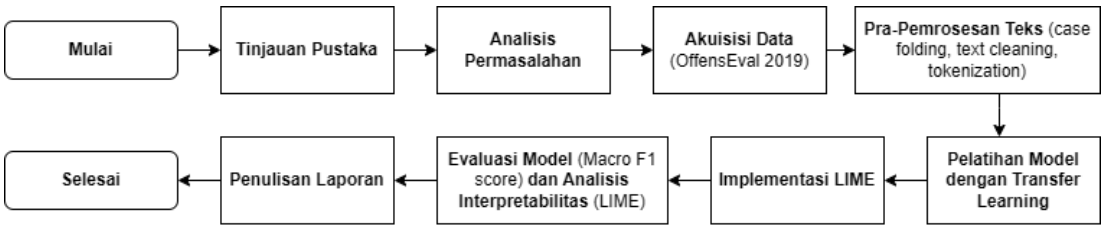
4.1 Deskripsi Umum Penelitian

Penelitian ini bertujuan untuk mengevaluasi performa model HateBERT dalam mendeteksi ujaran kebencian pada teks bahasa Inggris dengan menggunakan *transfer learning* serta menerapkan XAI menggunakan LIME untuk meningkatkan interpretabilitas model. Model HateBERT merupakan hasil *fine-tuning* dari BERT pada dataset yang mengandung ujaran kebencian di platform Reddit (RAL-E), yang bertujuan untuk meningkatkan pemahaman model terhadap bahasa yang menyinggung dan kasar dalam konteks *online* (Caselli dkk., 2021).

Dataset yang digunakan dalam penelitian ini adalah *Offensive Language Identification Dataset* (OLID) yang dikembangkan dalam kompetisi SemEval-2019 Task 6: OffensEval atau OffensEval 2019, yang berisi 13.240 *tweet* berbahasa Inggris yang telah dianotasi secara manual menjadi beberapa kategori (Zampieri dkk., 2019). Penelitian ini berfokus pada *sub-task* A dari dataset OffensEval, yaitu mengklasifikasikan *tweet* menjadi *offensive* (OFF) atau *non-offensive* (NOT).

Rancangan penelitian ini melibatkan beberapa tahapan utama. Dataset yang digunakan akan diproses terlebih dahulu melalui tahap pra-pemrosesan teks, kemudian dilakukan tokenisasi menggunakan *tokenizer* HateBERT sebelum masuk ke tahap *training* model dengan *transfer learning*. Setelah model selesai dilatih, performanya akan dievaluasi menggunakan metrik seperti akurasi, *precision*, *recall*, macro F1-score, dan *POS F1-score*. Untuk meningkatkan transparansi dalam deteksi ujaran kebencian, metode XAI berupa LIME diterapkan guna memberikan interpretasi terhadap keputusan model.

Kerangka umum penelitian ini mencakup beberapa langkah utama, yang divisualisasikan pada Gambar 4.1. Langkah-langkah penelitian ini terdiri dari tinjauan pustaka, analisis permasalahan, akuisisi data, pra-pemrosesan data, pengembangan metode, implementasi, evaluasi hasil, dan penulisan laporan.



Gambar 4.1 Diagram alur rancangan penelitian yang diusulkan

4.2 Analisis Permasalahan

Deteksi ujaran kebencian telah menjadi fokus utama dalam penelitian NLP karena meningkatnya interaksi di media sosial. Pendekatan awal menggunakan metode *rule-based* yang mengandalkan daftar kata atau frasa *offensive*. Namun, pendekatan ini sering menghasilkan tingkat *false positive* yang tinggi karena tidak mampu menangkap konteks dan variasi bahasa yang kompleks (Fortuna dan Nunes, 2018).

Seiring perkembangan teknologi, metode *machine learning* seperti SVM dan Naïve Bayes *classifier* mulai diterapkan. Meskipun menunjukkan peningkatan kinerja, metode ini memerlukan rekayasa fitur yang ekstensif dan masih menghadapi tantangan dalam memahami konteks ujaran kebencian (Fortuna dan Nunes, 2018). Selanjutnya, model *deep learning* seperti CNN dan LSTM digunakan untuk menangkap pola yang lebih kompleks dalam teks. Penelitian oleh Mnassri dkk. (2022) menunjukkan bahwa kombinasi model-model ini dapat meningkatkan akurasi deteksi ujaran kebencian. Namun, pendekatan ini memerlukan sumber daya komputasi yang besar dan kurang transparan dalam pengambilan keputusan.

Untuk mengatasi keterbatasan tersebut, Caselli dkk. (2021) mengembangkan HateBERT, model BERT yang dilatih ulang pada dataset khusus yang mengandung ujaran kebencian. HateBERT menunjukkan peningkatan akurasi dalam mendeteksi ujaran kebencian dibandingkan model BERT standar. Namun, meskipun kinerjanya meningkat, model ini tetap berperan sebagai "*black box*" yang sulit dipahami proses pengambilan keputusannya.

Pentingnya interpretabilitas dalam model AI mendorong pengembangan teknik XAI. Ribeiro dkk. (2016) memperkenalkan LIME, sebuah metode untuk menjelaskan prediksi model dengan cara yang dapat dipahami manusia. Penerapan

LIME dalam konteks deteksi ujaran kebencian memungkinkan pengguna dan peneliti memahami alasan di balik klasifikasi tertentu, meningkatkan kepercayaan dan akuntabilitas sistem.

Meskipun berbagai pendekatan telah diusulkan, tantangan utama tetap pada pengembangan model yang tidak hanya akurat tetapi juga dapat dijelaskan. Penelitian ini bertujuan untuk menggabungkan keunggulan HateBERT dalam memahami ujaran kebencian dengan kemampuan interpretabilitas dari LIME. Dengan menerapkan *transfer learning* pada HateBERT menggunakan dataset OffensEval 2019 dan mengintegrasikan LIME sebagai alat XAI, diharapkan dapat dicapai model yang efektif dan transparan dalam mendeteksi serta menjelaskan keputusan terkait ujaran kebencian.

4.3 Akuisisi dan Pengumpulan Data

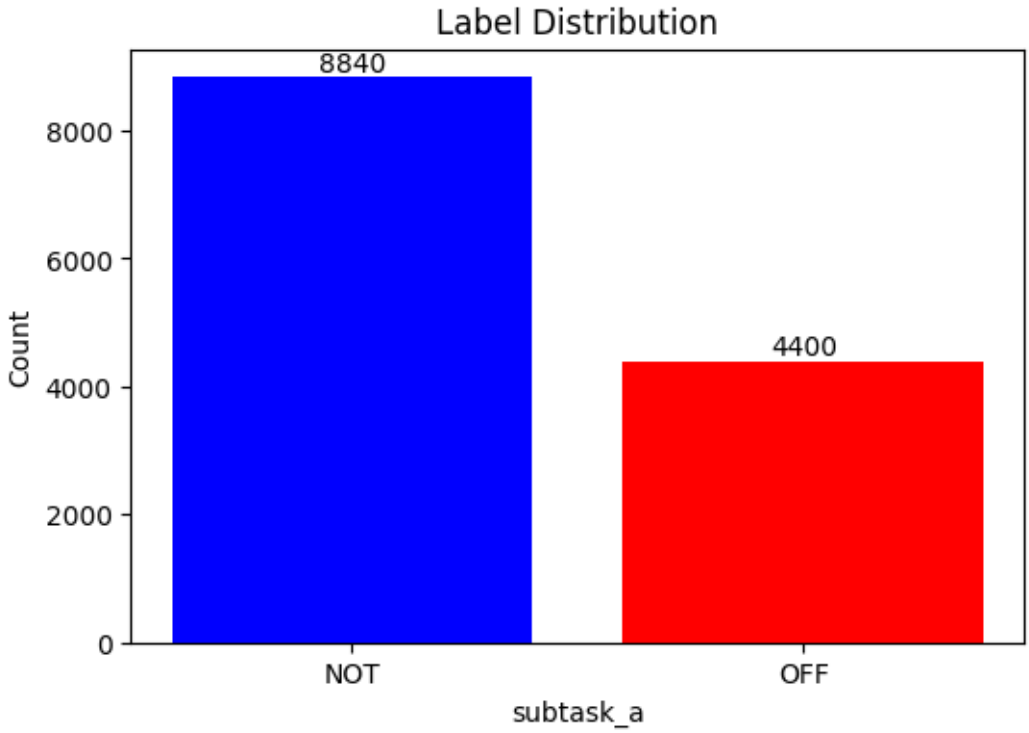
Dataset yang digunakan dalam penelitian ini adalah OffensEval 2019, yang dikembangkan dalam kompetisi SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (Zampieri dkk., 2019). Dataset ini terdiri dari 13.240 *tweet* berbahasa Inggris yang telah dianotasi secara manual untuk mengklasifikasikan ujaran kebencian dalam media sosial. Anotasi pada OffensEval 2019 dibagi menjadi tiga *sub-task*, yaitu *sub-task* A yang mengklasifikasikan *tweet* menjadi *offensive* (OFF) atau *non-offensive* (NOT), *sub-task* B yang mengategorikan apakah ujaran kebencian ditargetkan atau tidak, serta *sub-task* C yang menentukan target dari ujaran kebencian, apakah ditujukan kepada individu, kelompok, atau kategori lainnya.

Dalam penelitian ini, hanya *sub-task* A yang digunakan, dengan setiap *tweet* diklasifikasikan sebagai OFF atau NOT, yang kemudian dikonversi ke format numerik dengan 1 untuk OFF dan 0 untuk NOT agar dapat digunakan dalam proses pelatihan model. Dataset OffensEval 2019 dipilih karena telah digunakan secara luas dalam penelitian deteksi ujaran kebencian, serta mencakup anotasi yang memungkinkan pengujian model berbasis *transfer learning* seperti HateBERT. Tabel 4.1 memperlihatkan contoh potongan dataset OffensEval 2019 yang digunakan dalam penelitian ini.

Tabel 4.1 Potongan dataset OffensEval 2019

id	*tweet*	*subtask_a*	*subtask_b*	*subtask_c*
86426	*@USER She should ask a few native Americans what their take on this is.*	OFF	UNT	NaN
90194	*@USER @USER Go home you’re drunk!!! @USER #MAGA #Trump2020 🤮🇺🇸🤮 URL*	OFF	TIN	IND
16820	*Amazon is investigating Chinese employees who are selling internal data to third-party sellers looking for an edge in the competitive marketplace. URL #Amazon #MAGA #KAG #CHINA #TCOT*	NOT	NaN	NaN
62688	*@USER Someone should'veTaken" this piece of shit to a volcano. 😂"*	OFF	UNT	NaN
43605	*@USER @USER Obama wanted liberals & illegals to move into red states*	NOT	NaN	NaN

Pada Gambar 4.2 ditampilkan pembagian kelas dalam dataset berdasarkan label *sub-task a*, yaitu antara kategori OFF dan NOT. Visualisasi ini menunjukkan bahwa terdapat ketidakseimbangan kelas, dengan jumlah data dengan label NOT lebih banyak dibandingkan dengan data berlabel OFF. Jumlah data untuk label NOT adalah 8840, sedangkan untuk label OFF adalah 4400.



Gambar 4.2 Distribusi *subtask_a* NOT dan OFF pada dataset OffensEval 2019

4.4 Rancangan Algoritma Metode

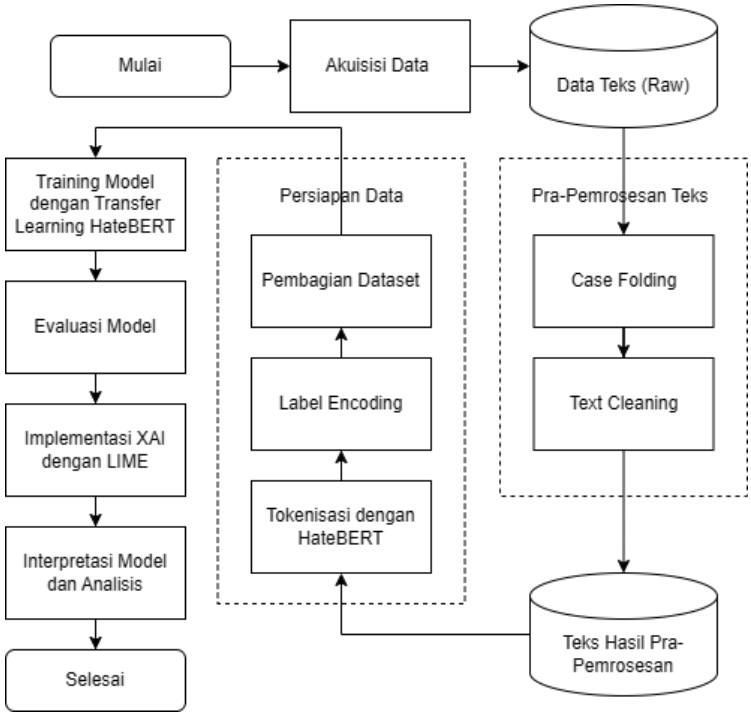
4.4.1 Gambaran Umum

Algoritma metode penelitian untuk deteksi ujaran kebencian menggunakan HateBERT dan XAI berupa LIME diperlihatkan pada Gambar 4.3. Penelitian ini dimulai dengan mengakuisisi dataset OffensEval 2019, yang berisi kumpulan *tweet* berbahasa Inggris yang telah dianotasi untuk deteksi dan klasifikasi ujaran kebencian (Zampieri dkk., 2019). Dataset yang diperoleh akan melalui tahapan pra-pemrosesan teks, yang mencakup *case folding*, *text cleaning*, dan *label encoding*, untuk memastikan konsistensi data sebelum masuk ke tahap tokenisasi.

Setelah pra-pemrosesan selesai, teks akan dikonversi menjadi token menggunakan *tokenizer* HateBERT, yang dirancang untuk menangani ujaran kebencian dalam Bahasa Inggris. Token yang dihasilkan akan digunakan dalam tahap training model dengan *transfer learning*, dengan melakukan *fine-tuning* mada model menggunakan dataset OffensEval 2019 untuk meningkatkan kemampuannya dalam mendeteksi ujaran kebencian.

Tahap berikutnya adalah evaluasi model dengan mengukur performa model menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score* untuk memastikan efektivitas model dalam mengklasifikasikan teks sebagai ujaran kebencian atau tidak. Setelah model dievaluasi, penelitian ini menerapkan XAI menggunakan LIME untuk memahami bagaimana model membuat keputusan dalam deteksi ujaran kebencian. Dengan menggunakan LIME, penelitian ini bertujuan untuk memberikan interpretasi terhadap prediksi model, sehingga meningkatkan transparansi dalam deteksi ujaran kebencian otomatis.

Gambar 4.3 memperlihatkan keseluruhan alur penelitian ini, mulai dari akuisisi data, pra-pemrosesan, tokenisasi, pelatihan model, evaluasi performa, hingga implementasi XAI dengan LIME dan analisis interpretabilitas model.



Gambar 4.3 Diagram rancangan algoritma metode yang diusulkan

4.4.2 Pra-Pemrosesan Teks

Pra-pemrosesan teks merupakan tahap awal yang dilakukan untuk memastikan bahwa data yang digunakan dalam penelitian ini memiliki format yang sesuai sebelum diproses lebih lanjut. Pada penelitian ini, diterapkan beberapa teknik pra-pemrosesan yang bertujuan untuk menyederhanakan teks serta menghilangkan elemen yang tidak diperlukan agar lebih sesuai dengan proses selanjutnya.

Tahapan pra-pemrosesan yang digunakan dalam penelitian ini terdiri dari *case folding* dan *text cleaning*. *Case folding* dilakukan untuk menyeragamkan bentuk teks, sementara *text cleaning* bertujuan untuk menghapus elemen-elemen yang tidak relevan. Selain itu, dataset juga mengalami proses penghapusan fitur yang tidak digunakan dalam penelitian ini, yaitu ID (diberikan ID baru), *subtask-b*, dan *subtask-c*, karena penelitian hanya berfokus pada *subtask-a*. Dengan adanya tahap pra-pemrosesan ini, teks dalam dataset telah dipersiapkan agar lebih terstruktur untuk tahap selanjutnya yang akan dibahas lebih lanjut dalam sub-bab berikutnya.

4.4.2.1 Case Folding

Case folding adalah proses dalam pra-pemrosesan teks yang bertujuan untuk menyeragamkan penggunaan huruf dengan mengubah semua karakter dalam teks menjadi huruf kecil (*lowercase*). Langkah ini penting untuk memastikan konsistensi data, sehingga model tidak membedakan antara kata yang sama dengan penggunaan huruf besar atau kecil yang berbeda (Aggarwal, 2018). Misalnya, kata "Saya", "SAYA", dan "saya" akan dianggap identik setelah proses *case folding*.

Penerapan *case folding* membantu dalam mengurangi kompleksitas data dan meningkatkan akurasi model dalam memproses teks. Dengan menghilangkan perbedaan yang disebabkan oleh huruf kapital, model dapat lebih fokus pada pola dan makna yang terkandung dalam teks. Dengan demikian, *case folding* merupakan langkah awal yang penting dalam pra-pemrosesan teks untuk memastikan data yang konsisten dan siap untuk tahap analisis selanjutnya.

4.4.2.2 Text Cleaning

Text cleaning merupakan salah satu tahap penting dalam pra-pemrosesan teks yang bertujuan untuk menghilangkan elemen-elemen yang tidak relevan atau dapat mengganggu performa model NLP. Data mentah dari media sosial sering kali mengandung berbagai bentuk *noise*, seperti *mention* (@user), URL, *hashtag*, emoji, serta penggunaan tanda baca yang tidak beraturan. Keberadaan elemen-elemen ini dapat menyebabkan model kesulitan dalam memahami makna sebenarnya dari teks (Liu dkk., 2005).

Dalam penelitian ini, *text cleaning* dilakukan dengan menghapus *mention* yang tidak relevan dalam deteksi ujaran kebencian. Teks yang mengandung kata "URL" sebagai representasi dari tautan dihapus sepenuhnya. *Hashtag* tetap dipertahankan dengan penghapusan simbol "#" agar kata dalam *hashtag* dapat diproses sebagai bagian dari teks tanpa kehilangan maknanya. Selain itu, *contractions* seperti "don't" atau "isn't" dikembangkan menjadi bentuk lengkap seperti "do not" dan "is not" agar model dapat mengenali bentuk kata secara utuh, yang penting dalam konteks klasifikasi ujaran kebencian. Emoji tidak dihapus, melainkan digantikan dengan deskripsi teks yang merepresentasikan maknanya, seperti emoji "😠" yang diubah menjadi :angry_face:. Pendekatan ini memungkinkan model untuk tetap menangkap konteks emosional dari teks tanpa perlu memproses karakter emoji secara langsung. Tanda baca tetap dipertahankan karena dapat memberikan informasi sintaksis penting dalam memahami struktur kalimat.

Dengan diterapkannya *text cleaning*, teks dalam dataset menjadi lebih bersih dan konsisten, sehingga memudahkan model dalam mengenali pola-pola ujaran kebencian secara lebih akurat pada tahap pemrosesan selanjutnya.

4.4.3 Persiapan Data

Setelah melalui tahap pra-pemrosesan, data perlu dipersiapkan lebih lanjut agar dapat digunakan dalam pelatihan model. Persiapan data merupakan tahap yang bertujuan untuk mengubah teks yang telah dibersihkan menjadi representasi numerik yang dapat dipahami model *deep learning*. Berbeda dengan pra-pemrosesan yang fokus pada pembersihan teks, persiapan data lebih fokus pada proses transformasi data agar sesuai dengan kebutuhan model yang akan digunakan (Young dkk., 2018).

Dalam penelitian ini, persiapan data dilakukan melalui beberapa tahapan utama. Diawali dengan tokenisasi menggunakan HateBERT yang bertujuan untuk mengubah teks menjadi token numerik sehingga dapat diproses oleh model berbasis *transformer*. Setelah itu, dilakukan *label encoding* dengan mengubah label kategorikal menjadi label numerik agar dapat digunakan dalam proses *training* model. Langkah terakhir dalam persiapan data adalah pembagian dataset menjadi

training set dan *validation set* untuk memastikan bahwa model dapat belajar dengan baik dari data latih dan diuji pada data validasi sebelum proses evaluasi akhir.

4.4.3.1 Tokenisasi Menggunakan HateBERT

Teks yang terdapat dalam media sosial sering kali memiliki struktur yang tidak baku, dengan penggunaan *slang*, kata-kata singkatan, serta ejaan yang tidak sesuai. Dalam konteks deteksi ujaran kebencian, tantangan ini semakin kompleks karena penggunaan bahasa sering kali bersifat kontekstual. Oleh karena itu, sebelum teks dapat diproses lebih lanjut oleh model NLP, diperlukan untuk mengubah teks mentah menjadi format yang lebih terstruktur dan dapat dipahami oleh model.

Dalam penelitian ini, transformasi tersebut dilakukan melalui tokenisasi berbasis *WordPiece*, yang merupakan bagian dari arsitektur HateBERT, sebuah model berbasis *transformer* yang telah disesuaikan untuk mendeteksi ujaran kebencian (Caselli dkk., 2021). HateBERT menggunakan metode *subword tokenization* melalui algoritma WordPiece, yang memungkinkan pemecahan kata menjadi unit yang lebih kecil berdasarkan frekuensi dalam korpus pelatihan. Teknik ini berguna dalam menangani kata-kata *out-of-vocabulary* (OOV) yang tidak ditemukan dalam kamus model. Sebagai contoh, kata "*unhappiness*" dapat dipecah menjadi token "[*un*]", "[*happiness*]" alih-alih dianggap sebagai kata yang sepenuhnya baru. Dengan pendekatan ini, model dapat menangani variasi morfologi kata secara lebih efisien dibandingkan tokenisasi berbasis *whitespace* atau karakter tunggal (Devlin dkk., 2019).

Dalam penelitian ini, teks yang telah melalui tahap *case folding* dan *text cleaning* dikonversi menjadi token menggunakan *tokenizer* dari HateBERT. Hasil tokenisasi terdiri dari beberapa komponen penting, yaitu *input IDs*, *attention masks*, dan t*oken type IDs*. *Input IDs* merupakan representasi numerik dari token yang dihasilkan, sementara *attention masks* digunakan untuk menandai token mana yang harus diproses oleh model. *Token type IDs* digunakan untuk membedakan segmen teks dalam model yang mendukung pasangan input.

Dengan diterapkannya tokenisasi ini, teks yang telah dibersihkan dapat diproses lebih lanjut dalam model *transformer*. Setelah tahap tokenisasi, data akan melalui proses *label encoding*, yang akan dibahas dalam sub-bab berikutnya.

4.4.3.2 Label Encoding

Model *machine learning* dan *deep learning* tidak dapat memahami label dalam bentuk teks secara langsung. Oleh karena itu, diperlukan proses *label encoding*, yaitu konversi label kategorikal menjadi format numerik agar dapat digunakan dalam pelatihan model (Müller dan Guido, 2016). Dalam penelitian ini, dataset yang digunakan memiliki label pada kolom *subtask_a*, yang mengklasifikasikan setiap *tweet* ke dalam dua kategori: "OFF" untuk ujaran kebencian dan "NOT" untuk bukan ujaran kebencian.

Agar label ini dapat digunakan dalam model, dilakukan *encoding* dengan aturan sebagai berikut: OFF dikonversi menjadi 1, yang menunjukkan bahwa teks tersebut mengandung ujaran kebencian, sedangkan NOT dikonversi menjadi 0, yang menunjukkan bahwa teks tidak mengandung ujaran kebencian. Dengan demikian, dataset yang sebelumnya berisi label dalam bentuk teks kini direpresentasikan dalam format numerik untuk memastikan kompatibilitas dengan model berbasis *deep learning*. Dengan proses ini, model dapat mengolah label secara lebih efisien dan menghasilkan prediksi yang lebih akurat.

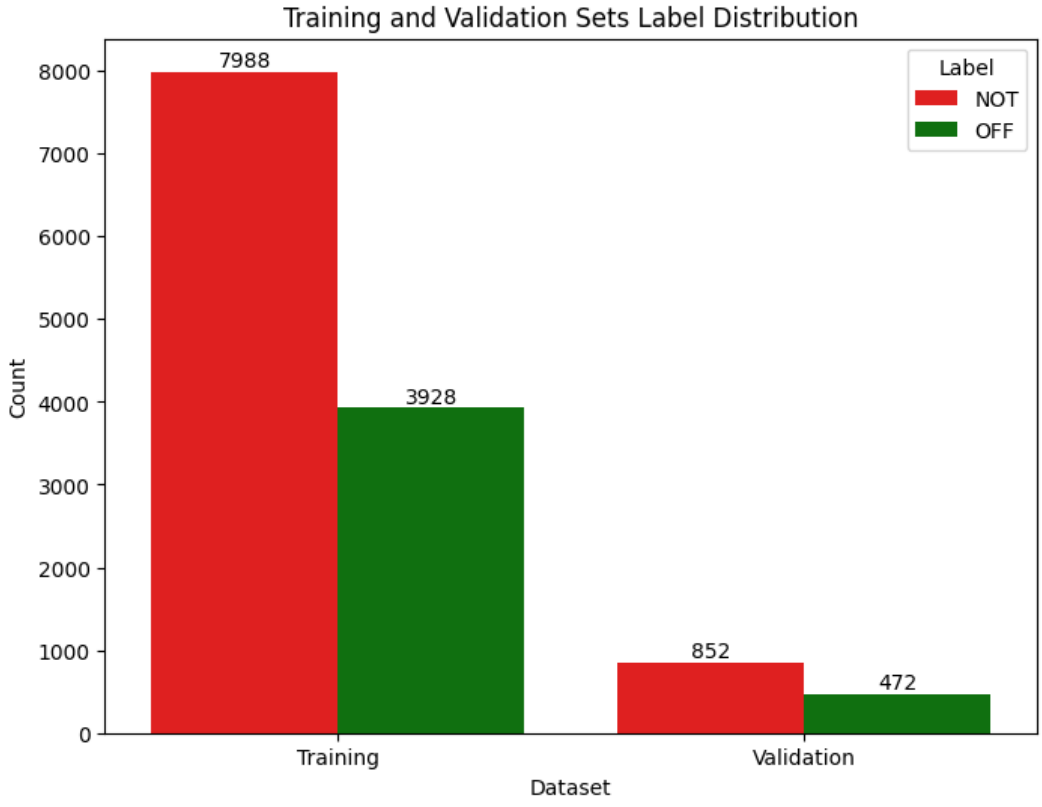
4.4.3.3 Pembagian Dataset

Pembagian dataset merupakan tahap penting dalam *machine learning* untuk memastikan bahwa model dapat belajar dari data yang cukup serta diuji pada data yang tidak pernah dilihat sebelumnya. Dalam penelitian ini, dataset dibagi menjadi dua bagian utama, yaitu *training set* dan *validation set* dengan rasio 90:10.

Sebanyak 90% dari total data digunakan sebagai *training set*, yang bertujuan untuk memberikan model eksposur yang maksimal terhadap berbagai variasi data, sehingga model dapat mempelajari pola ujaran kebencian secara lebih menyeluruh. Sementara itu, 10% sisanya digunakan sebagai *validation set* untuk menilai kemampuan generalisasi model selama proses pelatihan dan memastikan bahwa model tidak mengalami *overfitting* terhadap data *training set*. Rasio ini dipilih karena dengan jumlah data yang cukup besar, porsi 10% tetap dapat memberikan

hasil evaluasi yang baik dan juga memungkinkan pelatihan model dilakukan secara optimal menggunakan sebagian besar data (Muraina, 2022). Selain itu, pembagian ini sesuai untuk konteks model dengan kompleksitas tinggi dan tetap seperti HateBERT, dengan validasi tidak membutuhkan porsi yang besar untuk tetap memberikan estimasi performa yang representatif (Joseph, 2022).

Pada Gambar 4.4 ditampilkan grafik distribusi data berdasarkan label kelas dalam training set dan validation set. Grafik ini memperlihatkan bahwa baik training maupun validation set memiliki proporsi kelas yang tidak seimbang, di mana label NOT lebih mendominasi dibandingkan label OFF. Jumlah data pada training set adalah 11.916, terdiri dari 7.988 data berlabel NOT dan 3.928 data berlabel OFF. Sementara itu, validation set terdiri dari 1.324 data, dengan 852 data berlabel NOT dan 472 data berlabel OFF.



Gambar 4.4 Distribusi label kelas pada *training* dan *validation set*

4.5 Implementasi

Setelah dataset diproses dan dipersiapkan, tahap selanjutnya adalah implementasi metode yang telah dirancang. Implementasi dalam penelitian ini mencakup proses pelatihan model HateBERT menggunakan *transfer learning* serta penerapan XAI dengan metode LIME untuk menganalisis bagaimana model membuat keputusan dalam klasifikasi ujaran kebencian.

Pada tahap ini, model HateBERT yang telah ditokenisasi dengan dataset yang telah dibersihkan akan dilatih menggunakan dataset yang telah dibagi sebelumnya. Setelah model selesai dilatih, dilakukan evaluasi menggunakan metrik seperti *accuracy*, *precision*, *recall*, dan *F1-score*. Selain itu, penerapan XAI dilakukan untuk memberikan interpretasi terhadap hasil deteksi model dengan menganalisis kata-kata yang berkontribusi dalam keputusan prediksi.

4.5.1 Transfer Learning dengan HateBERT

Pada penelitian ini, model HateBERT digunakan sebagai dasar untuk deteksi ujaran kebencian dengan menerapkan *transfer learning*. HateBERT merupakan model berbasis BERT yang telah di-*train* ulang menggunakan data ujaran kebencian dalam bahasa Inggris, sehingga lebih efektif dalam mendeteksi dan memahami konteks ujaran kebencian dibandingkan model BERT standar (Caselli dkk., 2021).

Transfer learning diterapkan dengan memanfaatkan bobot awal dari HateBERT yang telah dilatih sebelumnya, kemudian dilakukan *fine-tuning* menggunakan dataset OffensEval 2019 yang telah diproses. Dengan pendekatan ini, model dapat menyesuaikan diri dengan dataset baru tanpa harus dilatih dari awal, sehingga mempercepat proses pelatihan sekaligus meningkatkan akurasi klasifikasi.

4.5.2 Implementasi Explainable AI dengan LIME

Dalam penelitian ini, metode LIME digunakan untuk memberikan gambaran terhadap keputusan model dalam deteksi ujaran kebencian. LIME merupakan teknik XAI yang bertujuan untuk memahami kontribusi setiap kata dalam sebuah teks terhadap prediksi model dengan membangun model *interpretable* di sekitar prediksi (Ribeiro dkk., 2016).

Implementasi LIME dilakukan dengan memilih sejumlah sampel teks dari dataset, kemudian menganalisis bagaimana model HateBERT memberikan prediksi terhadap teks tersebut. LIME bekerja dengan menghasilkan variasi dari teks asli dan mengevaluasi perubahan prediksi model, sehingga dapat mengidentifikasi kata-kata yang memiliki pengaruh signifikan dalam menentukan apakah sebuah teks termasuk dalam kategori ujaran kebencian atau tidak.

Hasil interpretasi LIME dapat divisualisasikan dalam bentuk *highlight* warna atau grafik, yang menunjukkan tingkat kontribusi setiap kata terhadap keputusan model. Dengan adanya analisis ini, model dapat lebih transparan dan dapat membantu dalam memahami bagaimana ujaran kebencian diklasifikasikan oleh model HateBERT.

4.6 Evaluasi Model dan Analisis Interpretabilitas

Evaluasi model dilakukan untuk mengukur performa klasifikasi ujaran kebencian setelah model HateBERT dilatih menggunakan *transfer learning*. Pengujian dilakukan dengan menghitung metrik evaluasi pada *validation set* dan testing set, yang telah dibagi dalam tahap sebelumnya. Selain itu, penerapan XAI dengan LIME digunakan untuk menganalisis interpretabilitas model dalam mengklasifikasikan teks sebagai ujaran kebencian atau bukan.

Proses evaluasi model dilakukan dengan menghitung metrik *accuracy*, *precision*, *recall*, dan *F1-score*, yang umum digunakan dalam klasifikasi teks untuk mengukur keseimbangan antara prediksi benar dan kesalahan model. *Accuracy* mengukur persentase prediksi yang benar dari keseluruhan data, sementara *precision* dan *recall* mengevaluasi sejauh mana model dapat mengidentifikasi ujaran kebencian dengan benar tanpa terlalu banyak kesalahan klasifikasi. *F1-score* digunakan sebagai ukuran keseimbangan antara *precision* dan *recall*, terutama pada dataset yang mungkin memiliki distribusi kelas yang tidak seimbang.

Selain evaluasi kuantitatif, penelitian ini juga menerapkan analisis interpretabilitas menggunakan LIME untuk memahami bagaimana model mengambil keputusan dalam klasifikasi teks. LIME bekerja dengan menganalisis kontribusi setiap kata dalam teks terhadap keputusan model, sehingga dapat memberikan wawasan mengenai fitur linguistik yang paling memengaruhi hasil

klasifikasi. Visualisasi hasil interpretasi ini digunakan untuk mengidentifikasi apakah model membuat keputusan berdasarkan kata-kata yang benar-benar relevan dengan ujaran kebencian atau justru terpengaruh oleh *bias* dalam data.

Evaluasi ini bertujuan untuk mengukur efektivitas *transfer learning* yang diterapkan pada HateBERT serta menilai transparansi model dalam menentukan prediksi. Dengan adanya kombinasi metrik evaluasi dan interpretabilitas, penelitian ini dapat memberikan pemahaman yang lebih mendalam mengenai kinerja model dalam mendeteksi ujaran kebencian serta faktor yang berperan dalam keputusan yang diambil oleh model.

4.7 Penulisan Laporan

Setelah penelitian selesai, seluruh proses yang dilakukan dalam pengembangan dan evaluasi model akan didokumentasikan dalam sebuah laporan. Laporan ini akan menjelaskan setiap tahapan penelitian, mulai dari pra-pemrosesan teks, penerapan *transfer learning* pada HateBERT, hingga analisis interpretabilitas menggunakan LIME. Selain itu, laporan juga akan dilengkapi dengan tabel-tabel yang menyajikan hasil evaluasi model menggunakan metrik *accuracy*, *precision*, *recall*, *macro F1-score*, dan *POS (Positive) F1-score*, serta visualisasi interpretasi model untuk memberikan gambaran yang lebih jelas mengenai proses deteksi ujaran kebencian yang dilakukan oleh model.

BAB V

IMPLEMENTASI

5.1 Lingkungan Implementasi

Pelaksanaan penelitian dilakukan dengan menggunakan dua lingkungan komputasi utama, yaitu komputer pribadi dan layanan komputasi awan dari Vast.ai. Komputer pribadi digunakan untuk eksplorasi awal, penulisan kode, serta pengujian skala kecil, sedangkan proses pelatihan model utama, termasuk fine-tuning HateBERT dan analisis interpretabilitas menggunakan LIME, dilakukan di lingkungan *cloud*. Selama seluruh proses penelitian, digunakan Jupyter Notebook sebagai lingkungan kerja interaktif, serta bahasa pemrograman Python sebagai bahasa utama dalam implementasi model, *data preprocessing*, *model training*, dan penerapan metode XAI.

Spesifikasi dari komputer pribadi yang digunakan selama proses pengembangan awal ditampilkan pada Tabel 5.1. Komputer ini menjalankan sistem operasi Windows dan memiliki spesifikasi perangkat keras yang mencukupi untuk keperluan penulisan kode, visualisasi data ringan, serta simulasi awal model sebelum dilakukan pelatihan penuh di lingkungan *cloud*.

Tabel 5.1 Spesifikasi dari komputer pribadi yang digunakan

Arsitektur	Spesifikasi
Sistem Operasi	Windows 11 Home
RAM	16 GB
CPU	AMD Ryzen 9 4900H with Radeon Graphics
GPU	NVIDIA Geforce RTX 2060 Laptop GPU

Untuk *model training* dan implementasi interpretabilitas model menggunakan LIME, digunakan layanan komputasi awan dari Vast.ai. Layanan ini menyediakan akses ke GPU dengan spesifikasi lebih tinggi dari berbagai penyedia independen. Selama pelaksanaan penelitian, digunakan beberapa jenis GPU.

Spesifikasi CPU dari masing-masing instans bersifat bervariasi tergantung ketersediaan penyedia, dan umumnya memiliki kapasitas yang lebih memadai untuk komputasi intensif. Rangkuman spesifikasi lingkungan komputasi awan ditampilkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi dari lingkungan Vast.ai

Arsitektur	Spesifikasi
GPU	RTX 3070, RTX 3070 TI, RTX 3080, RTX 3080 TI, dan RTX 4080S
CPU	Beragam tergantung instans

5.2 Inisialisasi Pustaka

Penelitian ini menggunakan berbagai pustaka Python yang mendukung proses analisis teks, pelatihan model berbasis BERT, serta interpretasi model menggunakan XAI. Seluruh pustaka yang diinisialisasi dapat dilihat pada Gambar 5.1.

Pustaka `pandas` dan `numpy` digunakan untuk manipulasi data, sedangkan `re`, `emoji`, dan `html` berperan penting dalam proses pembersihan teks seperti menghapus karakter khusus dan emoji, serta mengubah karakter HTML. Fungsi `train_test_split` dari modul `model_selection` dalam pustaka `scikit-learn` digunakan untuk membagi dataset menjadi data *training*, *validation*, dan *testing*, sementara `LabelEncoder` dari `preprocessing` digunakan untuk mengubah label kategorikal ke format numerik. Proses tokenisasi teks agar sesuai dengan input model BERT dilakukan menggunakan `BertTokenizer` dari pustaka `transformers`. Untuk visualisasi data dan hasil evaluasi model, digunakan pustaka `matplotlib.pyplot` dan `seaborn`. Pelatihan model dilakukan menggunakan pustaka `tensorflow`, yang terintegrasi dengan `TFBertForSequenceClassification` dari `transformers` sebagai model klasifikasi utama, serta `create_optimizer` untuk pengaturan algoritma optimisasi. Evaluasi performa model dilakukan menggunakan `classification_report`, `f1_score`, dan `confusion_matrix` dari

`sklearn.metrics`. Sementara itu, penerapan XAI menggunakan `LimeTextExplainer` dari `lime.lime_text` untuk memberikan interpretasi lokal terhadap hasil klasifikasi, dan modul `random` digunakan dalam pemilihan sampel data untuk proses interpretasi tersebut.

```
# Pustaka pengolahan dataset
import pandas as pd
import numpy as np
import re
import emoji
import html
import contractions
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from transformers import BertTokenizer
# Pustaka pelatihan model
import tensorflow as tf
import seaborn as sns
from transformers import TFBertForSequenceClassification,
create_optimizer, AdamWeightDecay
from sklearn.metrics import classification_report, f1_score,
confusion_matrix
# Pustaka implementasi LIME
import random
from lime.lime_text import LimeTextExplainer
```

Gambar 5.1 Kode untuk inisialisasi pustaka

5.3 Inisialisasi Dataset

Dataset yang digunakan dalam penelitian ini bersumber dari SemEval-2019 Task 6: OffensEval, yaitu kompetisi internasional yang bertujuan mengidentifikasi dan mengkategorikan ujaran ofensif dalam media sosial. Dataset ini diperkenalkan oleh Zampieri et al., (2019) dan terdiri dari lebih dari 13.240 *tweet* berbahasa Inggris yang telah dianotasi secara manual. Penelitian ini secara khusus memanfaatkan kolom *sub-task* A, yaitu klasifikasi biner terhadap *tweet* menjadi kategori OFF (offensive) atau NOT (non-offensive). Dataset disimpan dalam folder bernama dataset dan diberi nama OffensEval2019.xlsx.

Sebelum dilakukan pra-pemrosesan, langkah awal adalah membaca file dataset ke dalam memori dan menampilkan sebagian isi untuk eksplorasi struktur data. Gambar 5.2 merupakan kode untuk membaca dataset dan menampilkan lima baris pertama dari isi data sebagai gambaran awal.

```
# Membaca dataset dari file Excel dan menampilkan 5 baris awal
df = pd.read_excel("dataset/OffensEval2019.xlsx")
df.head()
```

Gambar 5.2 Kode untuk membaca dan menampilkan lima baris pertama dari dataset

Selanjutnya, untuk memahami distribusi jumlah data pada label *subtask_a*, dilakukan visualisasi perbandingan jumlah data yang termasuk kategori OFF dan NOT. Hal ini penting untuk mengetahui apakah dataset memiliki distribusi yang seimbang, karena ketidakseimbangan kelas dapat memengaruhi akurasi dan sensitivitas model klasifikasi. Gambar 5.3 merupakan kode yang digunakan untuk menampilkan grafik distribusi label *subtask_a*.

```
# Visualisasi distribusi label pada kolom subtask_a (OFF vs NOT)
plt.figure(figsize=(6,4))
df['subtask_a'].value_counts().plot(kind='bar', color=['tomato', 'lightgreen'])
plt.title("Distribusi Label subtask_a")
plt.xlabel("Kategori")
plt.ylabel("Jumlah Tweet")
plt.xticks(rotation=0)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

Gambar 5.3 Kode untuk visualisasi distribusi label pada *subtask_a*

5.4 Pra-pemrosesan dan Persiapan Data

Setelah dataset diinisialisasi, tahap berikutnya adalah melakukan pra-pemrosesan dan persiapan data. Tahapan ini penting untuk memastikan data yang digunakan dalam proses pelatihan model sudah dalam bentuk yang bersih dan siap untuk digunakan. Pada bagian pra-pemrosesan dilakukan *case folding* dan *text cleaning* yang meliputi penghilangan *hashtag*, *mention*, URL, serta konversi emoji menjadi teks deskriptif. Selain itu, dilakukan juga *feature selection* dengan memilih kolom-kolom yang relevan untuk analisis. Untuk persiapan data, dilakukan tokenisasi menggunakan HateBERT *tokenizer*, *encoding label* menjadi nilai numerik, serta pembagian dataset menjadi *subset training* dan *validation* untuk proses pelatihan dan evaluasi model.

Tahap awal pra-pemrosesan terlihat pada Gambar 5.4 yang memperlihatkan penghapusan kolom yang tidak relevan, penggantian nama kolom label, dan

penghapusan data yang kosong. Langkah ini bertujuan untuk menyederhanakan data dan memastikan kualitas data yang akan digunakan.

```
# Menghapus kolom yang tidak diperlukan
df = df.drop(columns=['id', 'subtask_b', 'subtask_c'])
df = df.rename(columns={'subtask_a': 'label'})
df = df.dropna()
df.head()
```

Gambar 5.4 Kode untuk *feature selection*

Selanjutnya, fungsi `clean_text` yang ditampilkan pada Gambar 5.5 berperan penting dalam membersihkan teks *tweet* dengan berbagai tahap pembersihan dan mengubah teks menjadi huruf kecil. Fungsi ini mengembangkan *contractions* seperti "*should've*" menjadi "*should have*", menghapus simbol *hashtag* atau tagar, *mention*, dan URL, serta mengubah emoji menjadi representasi teks menggunakan *library* `emoji`. Selain itu, fungsi ini juga menghilangkan spasi berlebih dan mengubah seluruh teks menjadi huruf kecil.

```
# Fungsi untuk membersihkan teks tweet
def clean_text(text):
 text = contractions.fix(text)
 text = re.sub(r'#', '', text)
 text = text.replace('@USER', '').replace('URL', '')
 text = emoji.demojize(text, delimiters=(":", ":"))
 text = re.sub(r'\s+', ' ', text)
 text = text.lower()
 return text.strip()
df['tweet'] = df['tweet'].apply(clean_text)
df[['tweet']].head()
```

Gambar 5.5 Kode fungsi *text cleaning* dan *case folding*

Setelah teks dibersihkan, langkah berikutnya adalah mengubah label kategori menjadi format numerik agar dapat diproses oleh model. Hal ini dilakukan menggunakan `LabelEncoder` dari pustaka `sklearn.preprocessing`, yang ditunjukkan pada Gambar 5.6. *Label encoding* ini mengonversi label “NOT” dan “OFF” masing-masing ke angka 0 dan 1.

```
# Menggunakan LabelEncoder untuk Label Encoding
label_encoder = LabelEncoder()
df['label'] = label_encoder.fit_transform(df['label'])
df[['label']].head()
```

Gambar 5.6 Kode untuk *label encoding*

Selanjutnya pada Gambar 5.7 memperlihatkan proses tokenisasi menggunakan HateBERT tokenizer dari pustaka transformers, yang menyiapkan

teks agar siap diinput ke model. Tokenisasi ini menerapkan *padding* dan *truncation* agar panjang input konsisten. Kemudian dataset dibagi menjadi data *training* dan *validation* menggunakan fungsi `train_test_split` dari `sklearn.model_selection` dengan proporsi 90% data *training* dan 10% data *validation*. Dataset hasil *split* juga disimpan dalam format .csv untuk keperluan pelatihan model berikutnya.

```
1   tokenizer =
    BertTokenizer.from_pretrained("GroNLP/hateBERT")
2   def tokenize_function(text):
3    return tokenizer(text, padding='max_length',
     truncation=True, max_length=128)
4   df_tokenized = df['tweet'].apply(tokenize_function)
5   tokenizer.save_pretrained('hatebert_tokenizer')
6   train_df, val_df = train_test_split(df, test_size=0.1,
    random_state=42)
7   train_df.to_csv('Dataset/offenseval_train.csv',
    index=False)
8   val_df.to_csv('Dataset/offenseval_val.csv', index=False)
9   print(f"Training Set Shape: {train_df.shape}")
10  print(f"Validation Set Shape: {val_df.shape}")
```

Gambar 5.7 Kode untuk tokenisasi dan pembagian dataset

5.5 Pembangunan dan Pelatihan Model

Setelah pra-pemrosesan dan data sudah disiapkan, tahap selanjutnya adalah membangun dan melatih model dengan menggunakan metode t*ransfer learning* HateBERT. Dataset *training* dan *validation* yang sudah disiapkan dimuat kembali dari file .csv dan diproses menjadi format yang dapat diterima oleh model. Proses tokenisasi dilakukan untuk menghasilkan *input IDs* dan *attention masks* sebagai fitur utama. Dataset ini kemudian dikonversi menjadi TensorFlow Dataset dengan pengacakan dan pembagian *batch* untuk mendukung pelatihan model secara efisien.

Penjelasan kode untuk memuat dataset dapat dilihat pada Gambar 5.8. Pada gambar ini memperlihatkan pembacaan file .csv untuk data *training* dan *validation* serta pemeriksaan awal data melalui tampilan beberapa baris pertama dan jumlah baris data yang tersedia.

```
train_data_path = 'Dataset/offenseval_train.csv'
val_data_path = 'Dataset/offenseval_val.csv'
train_df = pd.read_csv(train_data_path)
val_df = pd.read_csv(val_data_path)
train_df.head(), val_df.head()
print(f"Rows count in training dataset: {len(train_df)}")
print(f"Rows count in validation dataset: {len(val_df)}")
```

Gambar 5.8 Kode untuk memuat dataset

Pada Gambar 5.9 ditampilkan proses tokenisasi data training dan validasi menggunakan HateBERT tokenizer. Fungsi `tokenize_data` digunakan untuk mengubah teks menjadi *input IDs* dan *attention masks* yang diperlukan oleh model. Dataset ini kemudian dibungkus menjadi TensorFlow Dataset dengan *batch size* 32 dan data *training* diacak agar pelatihan menjadi lebih baik.

```
# Fungsi tokenisasi data
def tokenize_data(df):
 input_ids = []
 attention_masks = []
 for tweet in df['tweet']:
   encoded = tokenizer.encode_plus(
   tweet,
   add_special_tokens=True,
   max_length=100,
   padding='max_length',
   truncation=True,
   return_attention_mask=True,
   return_tensors='tf'
   )
  input_ids.append(encoded['input_ids'])
  attention_masks.append(encoded['attention_mask'])
 return np.concatenate(input_ids, axis=0),
 np.concatenate(attention_masks, axis=0)
train_input_ids, train_attention_masks =
tokenize_data(train_df)
val_input_ids, val_attention_masks = tokenize_data(val_df)
train_labels = train_df['label'].values
val_labels = val_df['label'].values
train_dataset = tf.data.Dataset.from_tensor_slices((
{'input_ids': train_input_ids, 'attention_mask':
train_attention_masks}, train_labels))
val_dataset = tf.data.Dataset.from_tensor_slices((
{'input_ids': val_input_ids, 'attention_mask':
val_attention_masks}, val_labels))
batch_size = 32
train_dataset =
train_dataset.shuffle(len(train_df)).batch(batch_size)
val_dataset = val_dataset.batch(batch_size)
```

Gambar 5.9 Kode untuk tokenisasi dan pembuatan dataset TensorFlow

Pada Gambar 5.10 diperlihatkan perhitungan bobot kelas (*class weight*) yang digunakan untuk menangani ketidakseimbangan data pada proses *training*. Selain itu, terdapat pengaturan *callback* penting yaitu `EarlyStopping` dengan `patience=5` yang menghentikan pelatihan apabila tidak ada peningkatan validasi selama 5 *epoch* berturut-turut, serta `ModelCheckpoint` yang menyimpan model terbaik berdasarkan akurasi validasi.

```
class_weights = compute_class_weight(
 'balanced',
 classes=np.unique(train_labels),
 y=train_labels
)
class_weight_dict = dict(zip(np.unique(train_labels),
class_weights))
print(f"Class Weights: {class_weight_dict}")
early_stopping = tf.keras.callbacks.EarlyStopping(
 monitor='val_accuracy',
 patience=5,
 restore_best_weights=True
)
model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
 "best_model.tf",
 monitor='val_accuracy',
 save_best_only=True,
 mode='max',
 verbose=1
)
```

Gambar 5.10 Kode untuk perhitungan *class weight* dan *callbacks* yang digunakan untuk *training*

Pada Gambar 5.11 mendefinisikan fungsi *loss*, *optimizer* Adam dengan *weight decay*, serta *learning rate scheduler* yang menerapkan *decay linear* terhadap *learning rat*e selama pelatihan. Jumlah *epoch* yang dideklarasikan adalah 25, namun karena callback EarlyStopping dengan `patience=5` diterapkan, pelatihan dapat berhenti lebih awal jika tidak ada perbaikan validasi. Model kemudian dikompilasi dan disiapkan untuk pelatihan. Akhirnya, model dilatih menggunakan fungsi `fit` dengan memasukkan dataset, *callback*, dan *parameter* terkait.

```
epochs = 25
steps_per_epoch = len(train_df) // batch_size
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
optimizer = AdamWeightDecay(
 learning_rate=1e-5,
 epsilon=1e-8,
 weight_decay_rate=0.1
)
def lr_scheduler(epoch, lr):
 total_steps = steps_per_epoch * epochs
 new_lr = lr * (1 - epoch / total_steps)
 return new_lr
lr_schedule = tf.keras.callbacks.LearningRateScheduler(lr_scheduler, verbose=1)
print("Dropout config:")
print(f"Hidden dropout prob: {model.config.hidden_dropout_prob}")
print(f"Attention probs dropout prob: {model.config.attention_probs_dropout_prob}")
model.compile(optimizer=optimizer, loss=loss_fn, metrics=['accuracy'])
model.summary()
history = model.fit(
 train_dataset,
 epochs=epochs,
 validation_data=val_dataset,
 callbacks=[model_checkpoint, early_stopping, lr_schedule],
 verbose=1
)
```

Gambar 5.11 Kode untuk *optimizer*, *learning rate scheduler*, kompilasi model, dan *model training*

5.6 Pengujian Model

Setelah model selesai dilatih, langkah berikutnya adalah melakukan pengujian untuk mengevaluasi performa model pada data *validation*. Pengujian ini meliputi analisis metrik evaluasi seperti *validation accuracy* dan berbagai varian *F1-score*, serta visualisasi kurva *training* dan *confusion matrix* untuk melihat sebaran prediksi model terhadap data aslinya.

Pada Gambar 5.12 ditampilkan kode untuk memvisualisasikan kurva *loss* dan *accuracy* selama proses *training* dan *validation*. Grafik ini memberikan gambaran tentang bagaimana performa model berubah pada tiap *epoch* saat *training*.

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

Gambar 5.12 Kode untuk visualisasi *loss* dan *accuracy* saat *training*

Selanjutnya pada Gambar 5.13 terlihat kode evaluasi performa model pada dataset validasi. Model dievaluasi dengan menghitung *loss* dan *accuracy*, serta melakukan prediksi untuk menghitung metrik seperti *validation accuracy*, *macro F1-score*, dan *POS F1-score*. *F1-score* merupakan ukuran yang menghubungkan *precision* dan *recall* yang memperlihatkan keseimbangan antara kedua metrik tersebut. *F1-score macro* menghitung *F1-score* secara rata-rata sederhana dari semua kelas tanpa memperhatikan proporsi kelas, sehingga cocok untuk dataset yang tidak seimbang. Sedangkan *F1-score POS* mengacu pada *F1-score* khusus untuk kelas positif, kelas ujaran kebencian, yang penting untuk menilai performa model dalam mendeteksi kasus positif. Hasil evaluasi ini memberikan gambaran komprehensif tentang kemampuan model dalam mendeteksi ujaran kebencian.

```
eval_results = model.evaluate(val_dataset, verbose=1)
print(f"Evaluation Results - Loss: {eval_results[0]}, Accuracy: {eval_results[1]}")
y_pred = model.predict(val_dataset)["logits"]
y_pred = tf.argmax(y_pred, axis=1).numpy()
val_accuracy = (y_pred == val_labels).mean()
val_macro_f1 = f1_score(val_labels, y_pred, average='macro')
val_pos_f1 = f1_score(val_labels, y_pred, average='binary')
print(f"Validation Accuracy: {val_accuracy:.4f}")
print(f"Validation macro F1-score: {val_macro_f1:.4f}")
print(f"Validation Positive F1-score (POS): {val_pos_f1:.4f}")
print("Classification Report (Validation):")
print(classification_report(val_labels, y_pred, digits=4))
```

Gambar 5.13 Kode untuk evaluasi model dan perhitungan metrik

Pada Gambar 5.14 disajikan kode untuk menampilkan *confusion matrix* dari hasil prediksi model pada data *validation*. Matriks ini divisualisasikan menggunakan *heatmap* untuk memperjelas distribusi prediksi benar dan salah pada tiap kelas, sehingga memudahkan interpretasi kesalahan klasifikasi model.

```
cm = confusion_matrix(val_labels, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['NOT', 'OFF'], yticklabels=['NOT', 'OFF'])
plt.title('Confusion Matrix (Validation)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```

Gambar 5.14 Kode untuk visualisasi *confusion matrix*

Terakhir, pada Gambar 5.15 merupakan kode untuk melakukan penyimpanan model yang telah dilatih agar dapat digunakan pada implementasi berikutnya dan interpretasi model tanpa perlu pelatihan ulang.

```
model_save_path = 'hatebert_trained_model'
model.save(model_save_path)
print(f"Model saved to {model_save_path}")
```

Gambar 5.15 Kode untuk menyimpan model

5.7 Implementasi Interpretasi Model

Sebagai kelanjutan dari evaluasi model, interpretasi hasil prediksi juga dilakukan untuk memahami bagaimana model mengambil keputusan. Untuk melakukan hal tersebut, diterapkan teknik Explainable Artificial Intelligence (XAI) menggunakan metode LIME. LIME dipilih karena kemampuannya memberikan

penjelasan lokal yang mudah dipahami terkait prediksi model, khususnya dalam konteks teks. Dengan LIME, dapat dilihat fitur kata mana yang paling berkontribusi terhadap keputusan klasifikasi model untuk sebuah contoh data.

Contoh data yang digunakan untuk interpretasi diambil secara acak dari dataset *validation*, sehingga representatif dalam menggambarkan perilaku model pada data yang belum pernah dilatih.

Pada Gambar 5.16 diperlihatkan kode untuk memuat kembali model dan *tokenizer* HateBERT yang telah disimpan sebelumnya. Langkah ini memastikan interpretasi menggunakan model final yang sama seperti saat evaluasi.

```
model_path = 'hatebert_trained_model/'
model = tf.keras.models.load_model(
 model_path,
  custom_objects={
  'AdamWeightDecay': AdamWeightDecay,
  'TFBertForSequenceClassification': tf.keras.Model
 }
)
tokenizer_path = 'hatebert_tokenizer/'
tokenizer = BertTokenizer.from_pretrained(tokenizer_path)
print("Model and tokenizer loaded successfully.")
```

Gambar 5.16 Kode untuk memuat model dan *tokenizer*

Pada Gambar 5.17 ditampilkan kode yang memuat dataset *validation* dan memilih satu data secara acak untuk dilakukan interpretasi. Data tersebut kemudian dicetak untuk diketahui *tweet* dan *label* aslinya sebelum dianalisis dengan LIME.

```
val_data_path = 'Dataset/offenseval_val.csv'
val_df = pd.read_csv(val_data_path)
print(val_df.head())
random_idx = np.random.randint(0, len(val_df))
random_row = val_df.iloc[random_idx]
tweet = random_row['tweet']
real_label = random_row['label']
print(f"Random Sample Tweet: {tweet}")
print(f"Real Label: {real_label}")
```

Gambar 5.17 Kode untuk memuat dataset *validation* dan pemilihan sampel secara acak

Pada Gambar 5.18 merupakan inisialisasi objek LIME *Text Explainer* dan fungsi prediksi model agar dapat digunakan oleh LIME. Fungsi ini melakukan tokenisasi input teks, menjalankan prediksi model, dan mengembalikan probabilitas prediksi yang dibutuhkan oleh LIME untuk membangun penjelasan.

```
explainer = LimeTextExplainer(class_names=["Non-Offensive",
"Offensive"])
def label_to_text(label):
 return "Non-Offensive" if label == 0 else "Offensive"
def predict_proba(texts):
 inputs = tokenizer(texts, padding=True, truncation=True,
 max_length=128, return_tensors='tf')
 input_dict = {key: inputs[key] for key in ['input_ids',
 'attention_mask', 'token_type_ids']}
 prediction = model.predict(input_dict)
 logits = prediction['logits']
 return tf.nn.softmax(logits, axis=-1).numpy()
```

Gambar 5.18 Kode inisialisasi LIME dan fungsi prediksi untuk LIME

Terakhir, pada Gambar 5.19 diperlihatkan kode proses prediksi label untuk *tweet* yang dipilih, serta pembuatan dan penampilan penjelasan lokal dari model menggunakan LIME. Penjelasan ini membantu memahami kata-kata mana yang paling berkontribusi pada keputusan klasifikasi model.

```
probs = predict_proba([tweet])
predicted_label = probs.argmax(axis=1)[0]
print(f"Chosen Tweet: {tweet}")
print(f"Real Label: {label_to_text(real_label)}")
print(f"Predicted Label: {label_to_text(predicted_label)}")
explanation = explainer.explain_instance(tweet,
predict_proba, num_features=10)
explanation.show_in_notebook(text=True)
```

Gambar 5.19 Kode untuk prediksi dan pembuatan penjelasan oleh LIME

BAB VI

HASIL DAN PEMBAHASAN

6.1 Hasil Pra-Pemrosesan

6.1.1 Hasil Pra-Pemrosesan Text Cleaning dan Case Folding

Proses pra-pemrosesan awal bertujuan untuk membersihkan teks dari komponen yang tidak relevan serta menyederhanakan struktur bahasa agar dapat diproses lebih optimal oleh model. Tahapan yang dilakukan mencakup penghapusan kolom tidak terpakai seperti *id*, *subtask_b*, dan *subtask_c*, serta pembersihan elemen-elemen yang umum terdapat pada *tweet* seperti *user mention* yang dalam hal ini adalah @USER, simbol "#" pada *hashtag*, dan *placeholder* URL. Selain itu, dilakukan normalisasi teks dengan menghapus karakter khusus, mengubah emoji menjadi representasi teks, menghapus spasi dan baris berlebih, serta mengubah seluruh karakter menjadi huruf kecil melalui proses *case folding*.

Teks hasil implementasi pra-pemrosesan yang disebutkan di atas ditampilkan pada Tabel 6.1 yang memperlihatkan perbandingan data sebelum dan sesudah proses pembersihan dilakukan. Tabel tersebut menunjukkan perubahan dari teks mentah yang semula mengandung banyak simbol dan *noise* menjadi teks yang bersih dan konsisten, sehingga dapat digunakan untuk tahap pemrosesan selanjutnya.

Tabel 6.1 Perbandingan teks sebelum dan sesudah dilakukan *text cleaning* dan *case folding*

Sebelum *text cleaning* dan *case folding*	**Sesudah *text cleaning* dan *case folding***
@USER She should ask a few native Americans what their take on this is.	*she should ask a few native americans what their take on this is*
@USER @USER Go home you're drunk!!! @USER #MAGA #Trump2020 👊🇺🇸👊 URL	*go home you are drunk maga trump2020 :oncoming_fist::united_states::oncoming_fist:*
Amazon is investigating Chinese employees who are selling internal data to third-party sellers looking for an edge in the competitive marketplace. URL #Amazon #MAGA #KAG #CHINA #TCOT	*amazon is investigating chinese employees who are selling internal data to thirdparty sellers looking for an edge in the competitive marketplace amazon maga kag china tcot*
@USER Someone should'veTaken" this piece of shit to a volcano. 😂 "	*someone shouldvetaken this piece of shit to a volcano :face_with_tears_of_joy:*
@USER @USER Obama wanted liberals & illegals to move into red states	*obama wanted liberals amp illegals to move into red states*

6.1.2 Hasil Pra-Pemrosesan Stopword Removal

Stopword removal merupakan tahap lanjutan dalam pra-pemrosesan teks yang bertujuan untuk menghilangkan kata-kata umum yang sering muncul namun memiliki kontribusi informasi yang rendah terhadap makna kalimat, seperti "*and*", "*the*", "*is*", dan sebagainya. Kata-kata ini umumnya tidak memiliki nilai klasifikatif tinggi karena muncul secara merata pada berbagai jenis teks. Dalam konteks deteksi ujaran kebencian, proses ini dilakukan dengan hati-hati, mengingat beberapa kata yang tergolong *stopword* dalam kamus umum bisa saja memiliki makna signifikan tergantung konteks sosial atau emosional yang menyertainya. Oleh karena itu,

meskipun *stopword removal* meningkatkan efisiensi pemodelan, terdapat potensi penghilangan informasi penting jika dilakukan secara agresif.

Perbandingan hasil sebelum dan sesudah proses *stopword removal* ditampilkan pada Tabel 6.2. Tabel tersebut memperlihatkan bagaimana elemen-elemen teks yang bersifat umum dan berfrekuensi tinggi telah dihapus untuk menyederhanakan masukan bagi model klasifikasi. Hasilnya menunjukkan teks yang lebih fokus pada kata-kata bermuatan makna yang berpotensi lebih kuat dalam mempengaruhi prediksi model.

Tabel 6.2 Perbandingan Teks Sebelum dan Sesudah Proses Stopword Removal

Teks sebelum dilakukan *stopword removal*	**Teks setelah dilakukan *stopword removal***
she should ask a few native americans what their take on this is	*ask native americans take*
go home you are drunk maga trump2020 :oncoming_fist::united_states::oncoming_fist:	*go home drunk maga trump2020 : oncoming_fist : :united_states : :oncoming_fist :*
amazon is investigating chinese employees who are selling internal data to thirdparty sellers looking for an edge in the competitive marketplace amazon maga kag china tcot	*amazon investigating chinese employees selling internal data thirdparty sellers looking edge competitive marketplace amazon maga kag china tcot*
someone shouldvetaken this piece of shit to a volcano :face_with_tears_of_joy:	*someone shouldvetaken piece shit volcano : face_with_tears_of_joy :*
obama wanted liberals amp illegals to move into red states	*obama wanted liberals amp illegals move red states*

6.1.3 Hasil Tokenisasi

Tokenisasi merupakan tahap penting dalam pemrosesan teks yang dilakukan dengan memecah kalimat menjadi satuan kecil yang disebut *token*. Dalam penelitian ini, digunakan *tokenizer* dari model HateBERT, yaitu model berbasis BERT yang telah dilatih secara khusus pada data mengandung ujaran kebencian. Tokenizer ini menerapkan metode WordPiece untuk membagi kata ke dalam sub-kata sehingga lebih mampu menangani kata-kata yang tidak dikenal atau *slang* yang umum ditemukan dalam teks media sosial.

Hasil tokenisasi ditampilkan pada Tabel 6.3 yang memperlihatkan transformasi dari teks asli menjadi *token* dan *input IDs* yang digunakan oleh model HateBERT. Pada tabel tersebut, kolom pertama menampilkan teks asli, kolom kedua berisi *token* hasil *tokenizer*, dan kolom ketiga menunjukkan *input IDs* numerik. *Token* dan *input ID* yang saling terkait diberi *highlight* hitam dengan teks putih, menunjukkan pasangan *token* dan *ID* yang sesuai. Selain itu, token berwarna merah adalah *special tokens* seperti [CLS] yang menandai awal input dan [SEP] yang menandai akhir input. Penandaan warna ini memudahkan identifikasi hubungan langsung antara token dan representasi numeriknya, serta menyoroti peran *special tokens* penting dalam arsitektur *transformer* untuk pemrosesan input yang tepat.

Tabel 6.3 Hasil tokenisasi menggunakan *tokenizer* HateBERT

Tweet	Tokens	Input_IDS
she should ask a few native americans what their take on this is	*she,should,ask,a,few,native,americans,what,their,take,on,this,is*	101,2016,2323,3198,1037,2261,3128,4841,2054,2037,2202,2006,2023,2003,102
go home you are drunk maga trump2020 :oncoming_fist::united_states::oncoming_fist	*go,home,you,are,drunk,mag,##a,trump,##20,##20,:,on,##coming,_,fist,:,:,united,_,states,:,:,on,##coming,_,fist,:*	101,2175,2188,2017,2024,7144,23848,2050,8398,11387,11387,1024,2006,18935,1035,7345,1024,1024,2142,1035,2163,1024,1024,2006,18935,1035,7345,1024,102
amazon is investigating chinese employees who are selling internal data to thirdparty sellers looking for an edge in the competitive marketplace amazon maga kag china tcot	*amazon,is,investigating,chinese,employees,who,are,selling,internal,data,to,third,##par,##ty,sellers,looking,for,an,edge,in,the,competitive,marketplace,amazon,mag,##a,ka,##g,china,tc,##ot*	101,9733,2003,11538,2822,5126,2040,2024,4855,4722,2951,2000,2353,19362,3723,19041,2559,2005,2019,3341,1999,1996,6975,18086,9733,23848,2050,10556,2290,2859,22975,4140,102
someone shouldvetaken this piece of shit to a volcano :face_with_tears_of_joy:	*someone,should,##vet,##ake,##n,this,piece,of,shit,to,a,volcano,:,face,_,with,_,tears,_,of,_,joy,:*	101,2619,2323,19510,13808,2078,2023,3538,1997,4485,2000,1037,12779,1024,2227,1035,2007,1035,4000,1035,1997,1035,6569,1024,102
obama wanted liberals amp illegals to move into red states	*obama,wanted,liberals,amp,illegal,##s,to,move,into,red,states*	101,8112,2359,13350,23713,6206,2015,2000,2693,2046,2417,2163,102

6.2 Pelatihan Model

6.2.1 Pelatihan Model Tanpa Pra-Pemrosesan

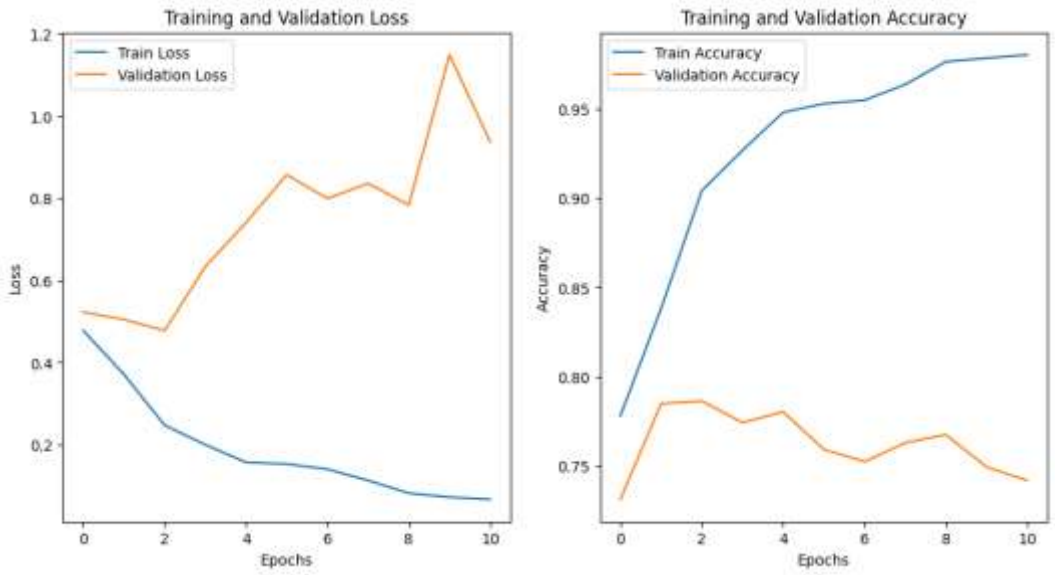
Pelatihan model dilakukan menggunakan data teks mentah tanpa melalui proses pra-pemrosesan. Model dilatih selama maksimum 25 *epoch* dengan penerapan metode *early stopping* menggunakan *parameter patience* 8 *epoch* untuk mencegah *overfitting*. Grafik *training* dan *validation loss* serta *accuracy* dapat

dilihat pada Gambar 6.1 untuk mengamati perkembangan performa model selama proses pelatihan.

Pada grafik tersebut, *training loss* secara konsisten menurun hingga mencapai nilai yang rendah, menunjukkan model mampu menyesuaikan diri dengan data latih. Namun, validation loss mulai meningkat setelah *epoch* ke-2, mengindikasikan model mengalami *overfitting*, dengan performa pada data *validation* tidak meningkat seiring dengan menurunnya *loss* pada data *training*. Hal ini juga tercermin pada grafik *accuracy*, dengan *training accuracy* mencapai nilai tertinggi sebesar 0.9804, sedangkan *validation accuracy* hanya mencapai 0.78625, menandakan adanya kesenjangan performa antara data *training* dan *validation*.

Proses pelatihan dihentikan pada *epoch* ke-11 karena *early stopping* yang memonitor tidak adanya perbaikan signifikan pada *validation loss* selama 8 *epoch* berturut-turut. Model menghasilkan F1-score sebesar 0.7593 pada data validasi, yang mencerminkan performa klasifikasi yang masih perlu ditingkatkan.

Polanya menunjukkan bahwa model mengalami *overfitting* cukup awal, yaitu setelah *epoch* ke-2, yang terlihat dari peningkatan *validation loss* meskipun *training loss* terus menurun. Hal ini wajar terjadi karena tanpa pra-pemrosesan, data teks mengandung banyak *noise*, sehingga model cenderung menghafal pola yang tidak dapat digeneralisasi. Dibandingkan dengan pelatihan berikutnya, model pada tahap ini menunjukkan jarak performa terbesar antara *training accuracy* dan *validation accuracy*, yang memperkuat indikasi *overfitting*.



Gambar 6.1 Grafik *training* dan *validation loss* serta *accuracy* model tanpa pra-pemrosesan

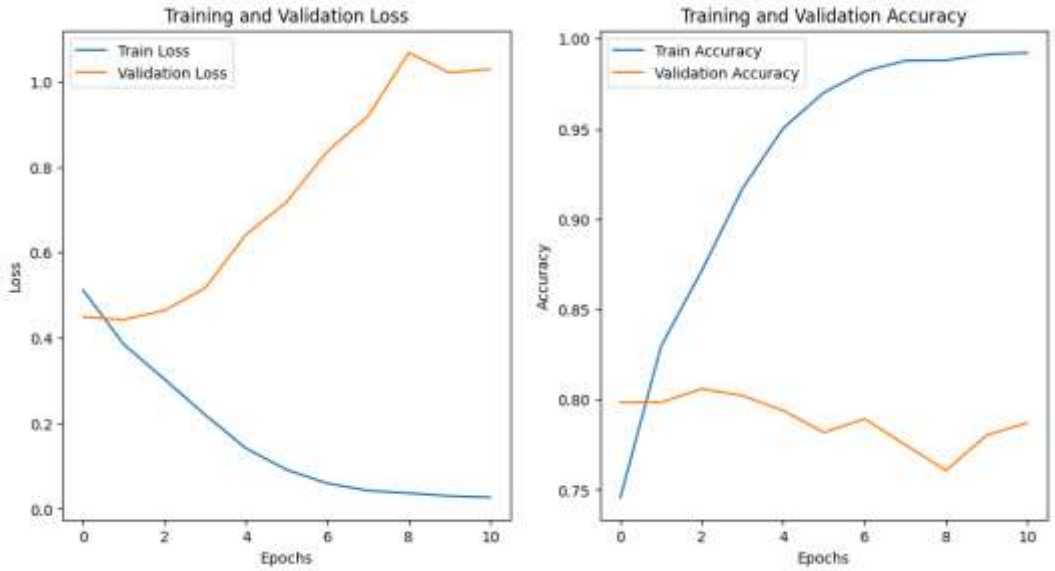
6.2.2 Pelatihan Model dengan Pra-Pemrosesan Dasar Berupa Text Cleaning dan Case Folding

Pada tahap ini, *model training* dilakukan dengan menggunakan data yang telah melalui pra-pemrosesan dasar, yakni *text cleaning* dan *case folding*. *Parameter training* dan metode *early stopping* tetap sama dengan *training* sebelumnya, dengan proses pelatihan berhenti pada *epoch* ke-11 karena tidak terjadi peningkatan signifikan pada *validation accuracy* selama 8 *epoch* berturut-turut. Perbandingan dengan pelatihan model tanpa pra-pemrosesan menunjukkan adanya peningkatan performa yang cukup signifikan.

Grafik *training* dan *validation loss* serta *accuracy* dapat dilihat pada Gambar 6.2. Pada grafik tersebut, *training loss* menunjukkan penurunan yang konsisten hingga mencapai nilai sangat rendah, sedangkan *validation loss* mengalami peningkatan setelah *epoch* ke-3 yang menunjukkan adanya *overfitting*. Grafik accuracy memperlihatkan training accuracy mencapai nilai tertinggi sebesar 0.9921, sedangkan *validation accuracy* tertinggi sebesar 0.8059, yang lebih baik dibandingkan pelatihan tanpa pra-pemrosesan. F1-score yang dihasilkan pada data *validation* sebesar 0.7878, menunjukkan peningkatan performa model dalam

klasifikasi setelah penerapan pra-pemrosesan dasar dibandingkan dengan pelatihan tanpa pra-pemrosesan.

Grafik menunjukkan bahwa pra-pemrosesan dasar membantu model untuk belajar lebih baik, terlihat dari *validation accuracy* yang lebih stabil dan lebih tinggi dibandingkan pelatihan tanpa pra-pemrosesan. Meskipun *overfitting* tetap terjadi setelah *epoch* ke-3, *validation loss* tidak meningkat secepat sebelumnya. Hal ini mengindikasikan bahwa pembersihan *noise* pada data membantu model fokus pada pola yang lebih relevan. Dibandingkan grafik pada Gambar 6.1, jarak antara *training* dan *validation accuracy* lebih kecil, menandakan peningkatan generalisasi, meskipun tidak secara signifikan.



Gambar 6.2 Grafik *training* dan *validation loss* serta *accuracy* model dengan pra-pemrosesan dasar

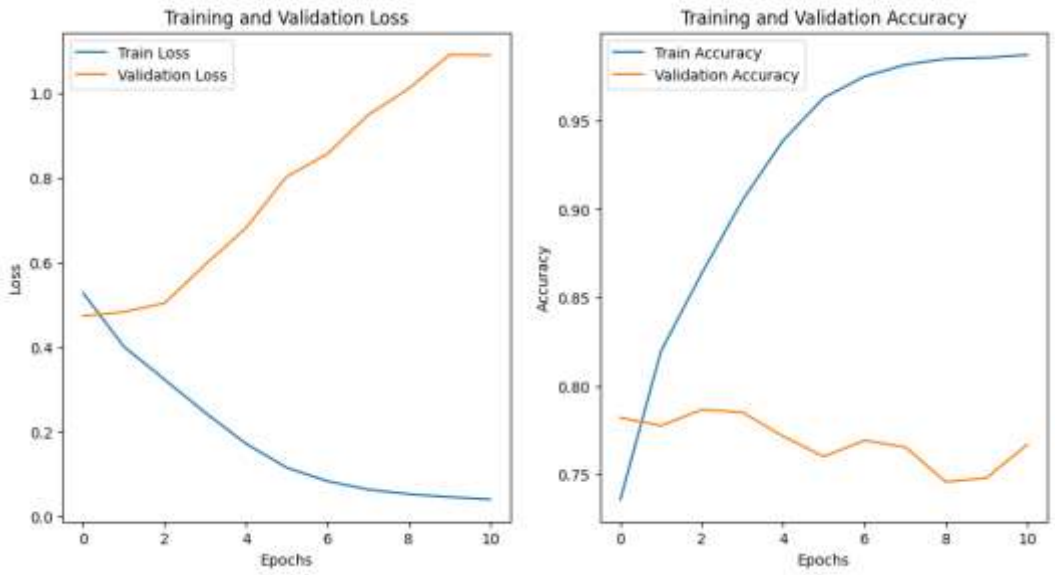
6.2.3 Pelatihan Model dengan Pra-Pemrosesan Dasar dan Stopword Removal

Pelatihan model dilakukan menggunakan data yang telah melalui pra-pemrosesan dasar serta tambahan tahap *stopword removal*. Proses pelatihan menggunakan *parameter* dan metode *early stopping* yang sama seperti sebelumnya, dengan pelatihan berhenti pada *epoch* ke-11 akibat tidak adanya peningkatan signifikan pada *validation accuracy* selama 8 *epoch* berturut-turut.

Grafik *training* dan *validation loss* serta *accuracy* dapat dilihat pada Gambar 6.3. Grafik tersebut menunjukkan penurunan *training loss* yang stabil hingga mencapai nilai sangat rendah, sementara *validation loss* mulai meningkat setelah *epoch* ke-2, menandakan terjadinya *overfitting*. *Training accuracy* mencapai nilai tertinggi sebesar 0.9871, dan *validation accuracy* tertinggi sebesar 0.7867. F1-score yang diperoleh pada data *validation* sebesar 0.7561, mencerminkan penurunan performa dibandingkan dengan pelatihan tanpa *stopword removal*.

Penurunan performa terlihat jelas pada grafik ini, dengan *validation loss* meningkat cukup tajam setelah *epoch* ke-2, dan *validation accuracy* cenderung lebih fluktuatif dibandingkan pada pra-pemrosesan dasar. Dibandingkan dengan Gambar 6.2, tren *validation accuracy* menjadi lebih tidak stabil, yang menunjukkan bahwa penghilangan *stopword* mengurangi kemampuan model dalam memahami konteks kalimat secara penuh. Hal ini konsisten dengan sifat HateBERT yang dilatih pada teks alami, yang berarti *stopword* membawa konteks penting dalam ujaran kebencian.

Jika dibandingkan dengan pelatihan menggunakan pra-pemrosesan dasar tanpa penghilangan *stopword*, hasil pelatihan kali ini menunjukkan performa yang sedikit menurun. Hal ini mengindikasikan bahwa penghilangan *stopword* dapat mengurangi efektivitas model dalam mempelajari konteks ujaran kebencian, sehingga *stopword* tetap memiliki peran penting dalam proses klasifikasi menggunakan HateBERT.



Gambar 6.3 Grafik *training* dan *validation loss* serta *accuracy* model dengan pra-pemrosesan dasar dan *stopword removal*

6.2.4 Hyperparameter Tuning

Proses *hyperparameter tuning* dilakukan dengan menguji beberapa nilai untuk *parameter* utama yang berpengaruh pada performa model. *Parameter* yang diuji meliputi *dropout layer*, *batch size*, *epoch*, *learning rate*, *epsilon*, dan *weight decay rate*. Nilai akhir yang dipilih dari hasil *tuning* untuk masing-masing *parameter* adalah sebagai berikut: *dropout layer* sebesar 0.5, *batch size* 32, *epoch* 25, *learning rate* 1e-5, *epsilon* 1e-8, dan *weight decay rate* 0.1.

Nilai *epoch* tetap diatur sebanyak 25 meskipun metode *early stopping* digunakan, karena nilai ini digunakan dalam perhitungan *total steps* pada *learning rate scheduler* yang mengatur penyesuaian *learning rate* selama pelatihan. *Learning rate*, *epsilon*, dan *weight decay rate* merupakan *parameter* utama pada *optimizer* AdamWeightDecay yang berperan dalam pengaturan proses optimasi *weight* model.

Tabel 6.4 menyajikan daftar nilai-nilai yang diuji untuk setiap *parameter*, dengan nilai akhir yang dipilih ditandai menggunakan cetak tebal. Tabel ini memberikan gambaran mengenai rentang nilai yang dipertimbangkan dalam proses *tuning* dan keputusan pemilihan *parameter* akhir yang digunakan dalam pelatihan model.

Tabel 6.4 Nilai *hyperparameter* yang dicoba

Hyperparameter	Nilai
Dropout layer	0.1, 0.2, **0.5**, 0.7
Batch size	16, **32**, 64, 128
Epoch	5, 10, 20, **25**, 50, 100
Learning rate	1e-6, 5e-5, 3e-5, **1e-5**, 5e-4, 1e-4
Epsilon	1e-9, **1e-8**, 1e-7, 1e-6, 1e-5
Weight decay rate	0.01, 0.02, 0.05, **0.1**, 0.2, 0.5

6.3 Pengujian dan Interpretasi Model

6.3.1 Evaluasi Pengaruh Pra-Pemrosesan terhadap Performa Model

Evaluasi performa model menggunakan beberapa metrik penting, yaitu *accuracy*, yang mengukur proporsi prediksi yang benar dari seluruh data; *precision*, yang menunjukkan ketepatan model dalam memprediksi kelas positif dengan menghitung persentase prediksi positif yang benar; dan *recall*, yang mengukur kemampuan model dalam mendeteksi seluruh contoh positif yang sebenarnya. Selain itu, digunakan *macro F1-score*, yaitu rata-rata harmonis *precision* dan *recall* dari masing-masing kelas tanpa mempertimbangkan jumlah data per kelas, sehingga menilai keseimbangan performa antar kelas. Sedangkan *POS F1-score* secara khusus mengukur performa model pada kelas target, yakni ujaran kebencian, yang menjadi fokus utama penelitian ini. Penggunaan kedua metrik ini penting untuk memastikan bahwa model tidak hanya unggul secara keseluruhan, tetapi juga efektif dalam mendeteksi kelas yang penting secara spesifik.

Berdasarkan hasil evaluasi, model dengan pra-pemrosesan dasar menunjukkan performa terbaik secara konsisten pada seluruh metrik, dengan *accuracy* tertinggi sebesar 0.8059 dan *macro F1-score* sebesar 0.7878. Perbandingan performa masing-masing model dapat dilihat pada Tabel 6.5. Sementara itu, penggunaan *stopword removal* justru menurunkan performa jika dibandingkan dengan pra-pemrosesan dasar, yang mengindikasikan bahwa penghilangan *stopword* dapat mengurangi kemampuan model dalam memahami

konteks ujaran kebencian. Oleh karena itu, model dengan pra-pemrosesan dasar dipilih sebagai model terbaik dan akan digunakan untuk tahap interpretasi dengan metode XAI (LIME).

Meskipun demikian, seluruh model yang diuji menunjukkan indikasi *overfitting*, yang terlihat dari selisih cukup besar antara nilai *training* dan *validation accuracy*. Fenomena ini umum terjadi pada model berbasis BERT dan HateBERT, yang dikarenakan kompleksitas model serta ukuran dataset yang relatif terbatas.

Tabel 6.5 Perbandingan evaluasi model berdasarkan metode pra-pemrosesan

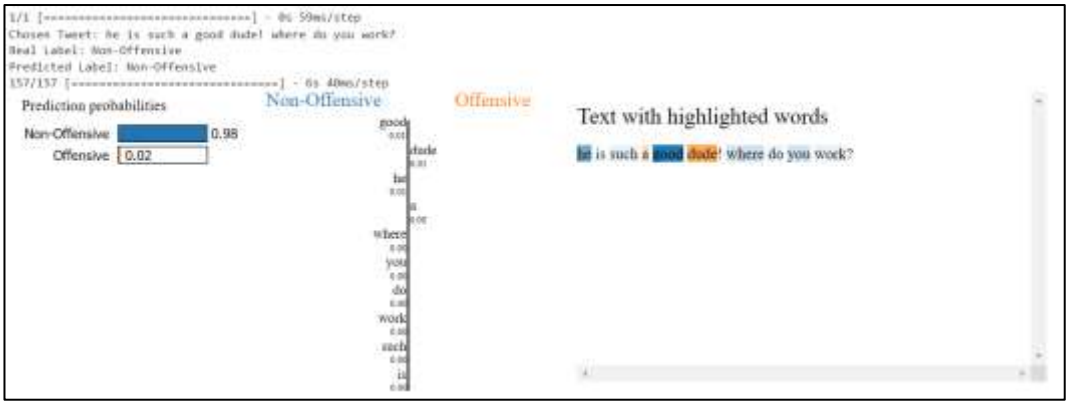
Metode Pra-Pemrosesan	***Accuracy***	***Precision***	***Recall***	***Macro F1-score***	***POS F1-score***
Tanpa pra-pemrosesan	0.7863	0.7825	0.7863	0.7593	0.6788
Pra-pemrosesan dasar	0.8059	0.8053	0.8059	0.7878	0.7257
Pra-pemrosesan dasar dan *stopword removal*	0.7867	0.7830	0.7867	0.7561	0.6698

6.3.2 Interpretasi Model dengan Pendekatan Explainable AI berupa LIME

Interpretasi model dilakukan menggunakan metode LIME. LIME merupakan pendekatan *post-hoc interpretability* yang menghasilkan penjelasan lokal terhadap prediksi model kompleks. Cara kerja LIME adalah dengan membangun model linier sederhana di sekitar satu *instance* (input teks) untuk mengestimasi pengaruh masing-masing fitur terhadap prediksi. Dalam konteks ini, fitur yang dimaksud adalah token kata pada input teks. Implementasi dilakukan dengan memanfaatkan pustaka `lime.lime_text`, dan interpretasi diberikan dalam bentuk visualisasi bobot kontribusi masing-masing kata terhadap kelas prediksi model.

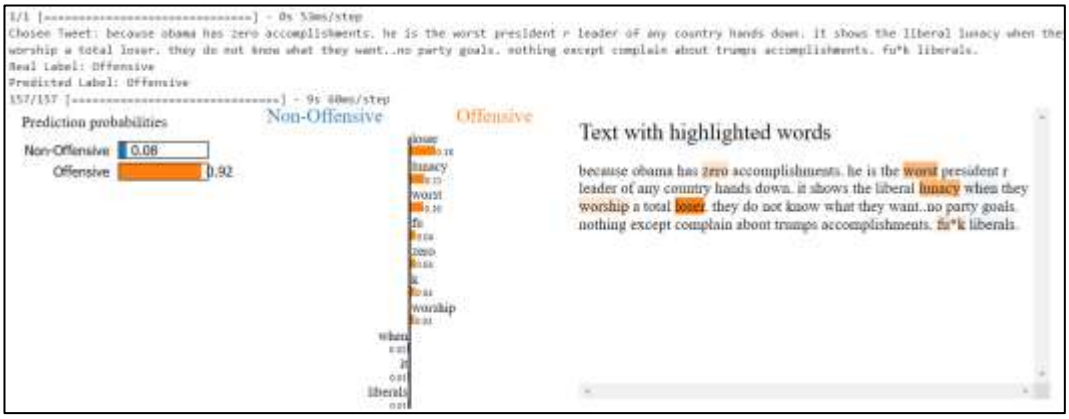
Contoh pertama merupakan salah satu data dari validation set yang diprediksi sebagai *Non-Offensive*. Model memberikan prediksi dengan probabilitas sebesar 0.98 terhadap kelas *Non-Offensive*. Pada visualisasi, kata "*he*" dan "*good*" ditandai sebagai kontribusi utama terhadap label tersebut, ditunjukkan oleh warna

biru. Kemungkinan besar, kata "*good*" dipahami oleh model sebagai bentuk pujian atau afirmasi positif yang secara umum tidak dikaitkan dengan ujaran kebencian dalam data pelatihan. Kata "*he*", sebagai kata ganti netral, juga banyak muncul dalam konteks netral atau sopan. Sebaliknya, kata "*dude*!" diberi warna jingga namun dengan bobot yang rendah, menandakan sedikit ketidakpastian yang dapat disebabkan karena sering muncul dalam berbagai konteks informal. Interpretasi ini ditampilkan pada Gambar 6.4.



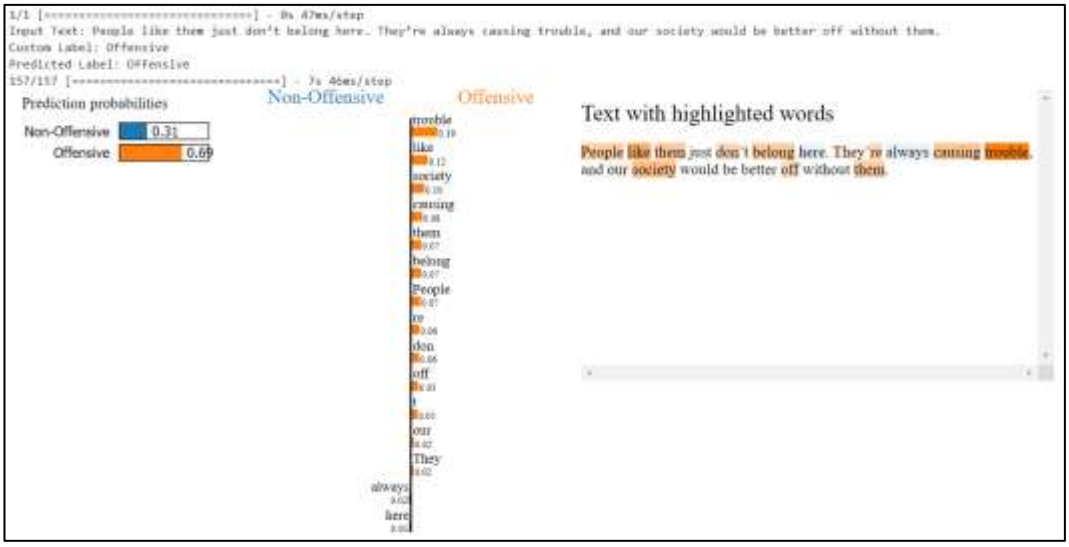
Gambar 6.4 Interpretasi LIME untuk teks *non-offensive* dari *validation set*

Contoh kedua juga berasal dari *validation set* dan diprediksi sebagai *Offensive* dengan probabilitas 0.92. Beberapa kata yang memiliki kontribusi tinggi terhadap klasifikasi ini antara lain "*loser*", "*worst*", "*lunacy*", dan "*fuck*". Warna jingga yang lebih tebal menunjukkan bahwa kata-kata tersebut berkontribusi tinggi dalam keputusan model. Kata-kata ini secara eksplisit mengandung makna negatif dan sering diasosiasikan dengan hinaan atau ekspresi kebencian dalam *training set*, sehingga pengaruhnya terhadap prediksi kelas *Offensive* menjadi dominan. Visualisasi interpretasi ini ditampilkan pada Gambar 6.5.



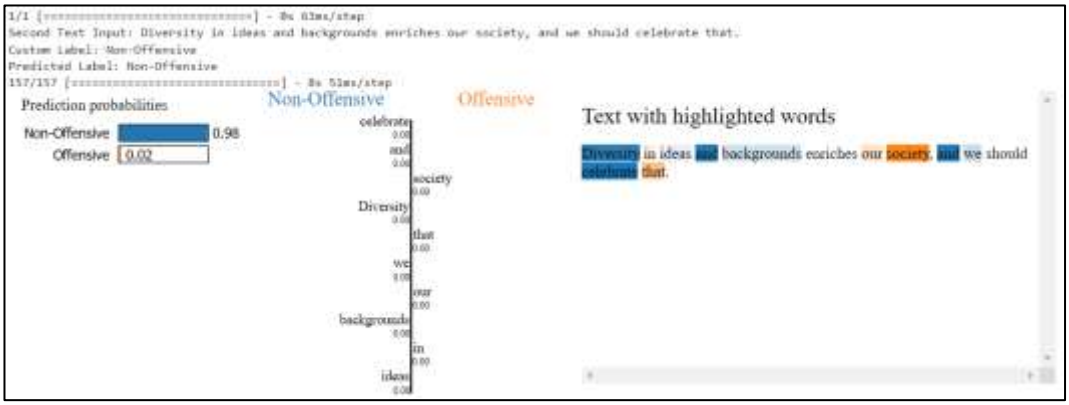
Gambar 6.5 Interpretasi LIME untuk teks *offensive* dari *validation set*

Contoh ketiga adalah input teks *offensive* buatan yang secara sengaja tidak mengandung kata kasar, namun menyiratkan ujaran kebencian terhadap kelompok tertentu. Model memprediksi teks tersebut sebagai *Offensive* dengan nilai *confidence* 0.69. Kata-kata seperti "*trouble*", "*like*", "*them*", dan "*society*" disorot sebagai kontribusi signifikan terhadap keputusan model. Kata-kata tersebut kerap muncul dalam kalimat yang dikaitkan dengan pemisahan kelompok atau stereotip negatif dalam *training set*, sehingga model mendeteksinya sebagai ujaran kebencian meskipun secara tidak vulgar. Interpretasi ini ditampilkan pada Gambar 6.6.



Gambar 6.6 Interpretasi LIME untuk teks buatan bersifat *offensive* tanpa kata kasar

Contoh keempat juga merupakan teks buatan namun bersifat *Non-Offensive*, meskipun memuat opini atau kritik sosial. Model memprediksi kelas *Non-Offensive* dengan nilai *confidence* tinggi sebesar 0.98. Kata-kata seperti "*diversity*", "*celebrate*", dan "*society*" berkontribusi memperkuat klasifikasi *Non-Offensive*. Kata-kata tersebut umumnya muncul dalam konteks inklusif dan positif dalam *training set*, dan cenderung tidak dikaitkan dengan ujaran kebencian. Hal ini menunjukkan bahwa model tidak hanya bergantung pada bentuk atau struktur kalimat, tetapi juga mampu memahami konteks nilai yang terkandung dalam ujaran. Hasil interpretasi ini ditampilkan pada Gambar 6.7.



Gambar 6.7 Interpretasi LIME untuk teks buatan bersifat non-*offensive* bertema kritik sosial

BAB VII

KESIMPULAN DAN SARAN

7.1 Kesimpulan

Berdasarkan hasil pelatihan model, pengujian performa, dan interpretasi prediksi yang telah dilakukan, diperoleh sejumlah poin kesimpulan yang merangkum temuan utama dari penelitian ini.

1. Model HateBERT yang telah dilatih ulang menggunakan dataset OffensEval 2019 mampu menjalankan tugas klasifikasi ujaran kebencian dengan performa yang baik. Setelah proses *transfer learning*, model yang menggunakan pra-pemrosesan dasar menghasilkan *accuracy* sebesar 0.8059, *macro F1-score* sebesar 0.7878, dan *POS F1-score* sebesar 0.7257. Hasil ini menunjukkan bahwa pelatihan ulang berhasil menyesuaikan model dengan karakteristik data baru, meskipun tidak secara signifikan meningkatkan skor dibanding model aslinya.
2. Variasi metode pra-pemrosesan memiliki dampak nyata terhadap performa model. Model dengan pra-pemrosesan dasar (*text cleaning* dan *case folding*) mendapat *accuracy* 0.8059, *macro F1-score* 0.7878, dan *POS F1-score* 0.7257. Tanpa pra-pemrosesan, model memperoleh *accuracy* 0.7863, *macro F1-score* 0.7593, dan *POS F1-score* 0.6788. Sementara itu, model dengan pra-pemrosesan dasar dan dengan tambahan *stopword removal* mencatat *accuracy* 0.7867, *macro F1-score* 0.7561, dan *POS F1-score* 0.6698. Penurunan performa saat menggunakan *stopword removal* mengindikasikan bahwa informasi penting bisa saja hilang jika *stopword* dihapus dalam konteks ujaran kebencian.
3. Jika dibandingkan dengan model HateBERT sebelum *transfer learning*, model yang asli mencatat *macro F1-score* 0.809 dan *POS F1-score* 0.723 pada data *pre-training*. Setelah dilakukan *transfer learning* dengan OffensEval, performa sedikit menurun dengan *macro F1-score* 0.7878, tetapi *POS F1-score* meningkat menjadi 0.7257. Dengan demikian, meskipun terdapat sedikit penurunan pada kinerja rata-rata antar kelas,

kemampuan model dalam mengenali kelas target (*offensive*) relatif tetap atau sedikit lebih baik.

4. Penerapan metode interpretasi model berbasis LIME membantu memberikan gambaran kontribusi kata terhadap prediksi, tetapi masih memiliki keterbatasan. Visualisasi menunjukkan kata-kata yang paling berkontribusi terhadap klasifikasi model, namun LIME hanya memberikan penjelasan lokal dan tidak selalu konsisten pada kasus yang serupa. Selain itu, metode ini tidak menjelaskan alasan semantik di balik kontribusi kata, sehingga tetap diperlukan intervensi manusia untuk menafsirkan hasil secara menyeluruh.
5. Dataset OffensEval 2019 memiliki struktur yang relatif bersih dan siap digunakan untuk pelatihan model deteksi ujaran kebencian. Hal ini terlihat dari hasil model tanpa pra-pemrosesan yang mencatat *accuracy* 0.7863, hanya sedikit lebih rendah dibanding model dengan pra-pemrosesan dasar dengan nilai *accuracy* 0.8059. Selisih yang tidak besar menunjukkan bahwa struktur data sudah cukup baik meskipun tanpa pembersihan tambahan.

7.2 Saran

Beberapa saran berikut dapat menjadi arah pengembangan lanjutan dari penelitian ini, baik untuk memperluas pendekatan maupun mengeksplorasi penerapan di bidang lain.

1. Menerapkan metode interpretasi yang memiliki kompleksitas lebih tinggi dan membutuhkan komputasi yang lebih besar, seperti SHAP atau Integrated Gradients, untuk memperoleh penjelasan yang lebih menyeluruh terhadap keputusan model.
2. Menerapkan pendekatan XAI pada kasus NLP lainnya seperti deteksi sentimen, berita palsu, atau klasifikasi emosi, agar pendekatan interpretasi dapat diuji pada konteks yang berbeda dan lebih bervariasi.
3. Menggabungkan model HateBERT dengan pendekatan *ensemble* untuk meningkatkan akurasi prediksi. Pendekatan ini berpotensi menghasilkan

model yang lebih kuat, namun membutuhkan konfigurasi sistem dan sumber daya komputasi yang lebih besar.

DAFTAR PUSTAKA

Aggarwal, C.C., 2018. Machine Learning for Text. [online] Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-73531-3.

Arras, L., Horn, F., Montavon, G., Müller, K.-R. and Samek, W., 2017. "What is relevant in a text document?": An interpretable machine learning approach. *PLOS ONE*, 12(8), p.e0181142. https://doi.org/10.1371/journal.pone.0181142.

Arrieta, A.B., Díaz-Rodríguez, N., Ser, J.D., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible *AI. Information Fusion*, 58, pp.82–115. https://doi.org/10.1016/j.inffus.2019.12.012.

Brown, A., 2017. What is hate speech? Part 1: The Myth of Hate. *Law and Philosophy*, 36(4), pp.419–468. https://doi.org/10.1007/s10982-017-9297-1.

Buhrmester, V., Münch, D. and Arens, M., 2019. Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey. https://doi.org/10.48550/arXiv.1911.12116.

Camacho-Collados, J. and Pilehvar, M.T., 2018. On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. In: T. Linzen, G. Chrupała and A. Alishahi, eds. *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. [online] Brussels, Belgium: Association for Computational Linguistics. pp.40–46. https://doi.org/10.18653/v1/W18-5406.

Caselli, T., Basile, V., Mitrović, J. and Granitzer, M., 2021. HateBERT: Retraining BERT for Abusive Language Detection in English. https://doi.org/10.48550/arXiv.2010.12472.

Davidson, T., Warmsley, D., Macy, M. and Weber, I., 2017. Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1), pp.512–515. https://doi.org/10.1609/icwsm.v11i1.14955.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: J. Burstein, C. Doran and T. Solorio, eds. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. [online] Minneapolis, Minnesota: Association for Computational Linguistics. pp.4171–4186. https://doi.org/10.18653/v1/N19-1423.

Feldman, R. and Sanger, J., 2006. The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press.

Fortuna, P. and Nunes, S., 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Comput. Surv.*, [online] 51(4). https://doi.org/10.1145/3232676.

Hare, I. and Weinstein, J. eds., 2009. Extreme Speech and Democracy. [online] Oxford University Press. https://doi.org/10.1093/acprof:oso/9780199548781.001.0001.

Hugging Face, 2023. Tokenizer. [online] Available at: <https://huggingface.co/docs/transformers/en/main_classes/tokenizer> [Accessed 26 February 2025].

Johannemann, J., Hadad, V., Athey, S. and Wager, S., 2021. Sufficient Representations for Categorical Variables. Available at: <https://arxiv.org/abs/1908.09874>.

Joseph, V.R., 2022. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(4), pp.531–538. https://doi.org/10.1002/sam.11583.

Jurafsky, D. and Martin, J., 2008. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

Kepolisian Negara Republik Indonesia, 2015. Surat Edaran No. SE/6/X/2015 tentang Ujaran Kebencian. [online] *Jakarta: Kepolisian Negara Republik Indonesia*. Available at: <https://pro.hukumonline.com/a/lt564076166fdd6/penanganan-ujaran-kebencian-hate-speech>.

Kibriya, H., Siddiqa, A., Khan, W.Z. and Khan, M.K., 2024. Towards safer online communities: Deep learning and explainable AI for hate speech detection and classification. *Computers and Electrical Engineering*, 116, p.109153. https://doi.org/10.1016/j.compeleceng.2024.109153.

Li, J., 2024. Causes of Action Identification in Disputes over Construction Project Contracts Using Hierarchical Learning Based on BERT. *Applied Mathematics and Nonlinear Sciences*, [online] 9(1). https://doi.org/10.2478/amns-2024-2254.

Liu, B., Hu, M. and Cheng, J., 2005. Opinion observer: analyzing and comparing opinions on the Web. In: *Proceedings of the 14th International Conference on World Wide Web*, WWW '05. [online] New York, NY, USA: Association for Computing Machinery. pp.342–351. https://doi.org/10.1145/1060745.1060797.

Lundberg, S.M. and Lee, S.-I., 2017. A Unified Approach to Interpreting Model Predictions. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds. *Advances in Neural Information Processing Systems*. [online] Curran Associates, Inc. Available at: <https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf>.

Manning, C.D. and Schütze, H., 1999. Foundations of Statistical Natural Language Processing. [online] Cambridge, Massachusetts: The MIT Press. Available at: <http://nlp.stanford.edu/fsnlp/>.

Mnassri, K., Rajapaksha, P., Farahbakhsh, R. and Crespi, N., 2022. BERT-based Ensemble Approaches for Hate Speech Detection. In: *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. pp.4649–4654. https://doi.org/10.1109/GLOBECOM48099.2022.10001325.

Molnar, C., 2022. Interpretable Machine Learning: A Guide for Making Black Box Models Explainable. 2nd ed. [online] Available at: <https://christophm.github.io/interpretable-ml-book>.

Mudde, C. and Rovira Kaltwasser, C., 2017. Populism: A Very Short Introduction. [online] Oxford University Press. https://doi.org/10.1093/actrade/9780190234874.001.0001.

Müller, A.C. and Guido, S., 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Inc.

Muraina, I., 2022. IDEAL DATASET SPLITTING RATIOS IN MACHINE LEARNING ALGORITHMS: GENERAL CONCERNS FOR DATA SCIENTISTS AND DATA ANALYSTS.

Murel, J. and Kavlakoglu, E., 2024. What is transfer learning? | *IBM*. [online] Available at: <https://www.ibm.com/topics/transfer-learning> [Accessed 3 December 2024].

Mutasodirin, M.A. and Prasojo, R.E., 2021. Investigating Text Shortening Strategy in BERT: Truncation vs Summarization. In: *2021 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. [online] IEEE. pp.1–5. https://doi.org/10.1109/icacsis53237.2021.9631364.

Pan, S.J. and Yang, Q., 2010. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10), pp.1345–1359. https://doi.org/10.1109/TKDE.2009.191.

Powers, D., 2011. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, 2(1), pp.37–63.

Reiter, E. and Dale, R., 2000. Building Natural Language Generation Systems. *Studies in Natural Language Processing*. Cambridge University Press.

Ribeiro, M.T., Singh, S. and Guestrin, C., 2016. 'Why Should I Trust You?': Explaining the Predictions of Any Classifier. Available at: <https://arxiv.org/abs/1602.04938>.

Ribeiro, M.T., Singh, S. and Guestrin, C., 2018. Anchors: High-Precision Model-Agnostic Explanations. *Proceedings of the AAAI Conference on Artificial Intelligence*, [online] 32(1). https://doi.org/10.1609/aaai.v32i1.11491.

Rodríguez, P., Bautista, M.A., Gonzàlez, J. and Escalera, S., 2018. Beyond one-hot encoding: Lower dimensional target embedding. *Image and Vision Computing*, 75, pp.21–31. https://doi.org/10.1016/j.imavis.2018.04.004.

Ruder, S., Peters, M.E., Swayamdipta, S. and Wolf, T., 2019. Transfer Learning in Natural Language Processing. In: A. Sarkar and M. Strube, eds. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. [online] Minneapolis, Minnesota: Association for Computational Linguistics. pp.15–18. https://doi.org/10.18653/v1/N19-5004.

Samek, W., Wiegand, T. and Müller, K.-R., 2017. Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models. Available at: <https://arxiv.org/abs/1708.08296>.

Schmidt, A. and Wiegand, M., 2017. A Survey on Hate Speech Detection using Natural Language Processing. In: *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*. [online] Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media. Valencia, Spain: Association for Computational Linguistics. pp.1–10. https://doi.org/10.18653/v1/W17-1101.

Schuster, M. and Nakajima, K., 2012. Japanese and Korean voice search. In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp.5149–5152. https://doi.org/10.1109/ICASSP.2012.6289079.

Song, X., Salcianu, A., Song, Y., Dopson, D. and Zhou, D., 2021. Fast WordPiece Tokenization. Available at: <https://arxiv.org/abs/2012.15524>.

Tjoa, E. and Guan, C., 2021. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11), pp.4793–4813. https://doi.org/10.1109/tnnls.2020.3027314.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. ukasz and Polosukhin, I., 2017. Attention is All you Need. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, eds. *Advances in Neural Information Processing Systems*. [online] Curran Associates, Inc. Available at: <https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf>.

Waldron, J., 2012. The harm in hate speech. Cambridge, MA, US: Harvard University Press. pp.vii, 292. https://doi.org/10.4159/harvard.9780674065086.

Wang, R., Wang, Z., Xu, Z., Wang, C., Li, Q., Zhang, Y. and Li, H., 2021. A Real-Time Object Detector for Autonomous Vehicles Based on YOLOv4. *Computational Intelligence and Neuroscience*, 2021, pp.1–11. https://doi.org/10.1155/2021/9218137.

Wiącek, M., Rybak, P., Pszenny, Ł. and Wróblewska, A., 2024. NLPre: A Revised Approach towards Language-centric Benchmarking of Natural Language Preprocessing Systems. In: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti and N. Xue, eds. *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. [online] Torino, Italia: ELRA and ICCL. pp.12271–12287. Available at: <https://aclanthology.org/2024.lrec-main.1073/>.

Young, T., Hazarika, D., Poria, S. and Cambria, E., 2018. *Recent Trends in Deep Learning Based Natural Language Processing*. Available at: <https://arxiv.org/abs/1708.02709>.

Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N. and Kumar, R., 2019. Predicting the Type and Target of Offensive Posts in Social Media. In: J. Burstein, C. Doran and T. Solorio, eds. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. [online] Minneapolis, Minnesota: Association for Computational Linguistics. pp.1415–1420. https://doi.org/10.18653/v1/N19-1144.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H. and He, Q., 2020. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, PP, pp.1–34. https://doi.org/10.1109/JPROC.2020.3004555.