BAB 2
LANDASAN TEORI

2.1 Studi Literatur

Penelitian mengenai deteksi ujaran kebencian (*hate speech detection*) telah berkembang secara signifikan seiring dengan meningkatnya peran media sosial sebagai ruang komunikasi publik. Karakteristik bahasa media sosial yang bersifat dinamis, tidak baku, serta sering melibatkan penggunaan bahasa campuran (*code-mixed*), kata gaul, singkatan, dan variasi ejaan, menjadikan pendeteksian ujaran kebencian sebagai permasalahan yang kompleks. Untuk mengatasi tantangan tersebut, berbagai pendekatan telah dikembangkan, mulai dari metode *deep learning* konvensional hingga pemanfaatan *pre-trained language models* berbasis arsitektur transformer. Rincian metrik dari penelitian-penelitian terkait terdapat pada Tabel 2.1.

Pada konteks teks berbahasa Inggris, sejumlah penelitian telah menunjukkan efektivitas model berbasis transformer dalam mendeteksi ujaran kebencian. Sukriti Narang et al. [14] memanfaatkan model BERT dengan penyetelan hyperparameter untuk mendeteksi ujaran kebencian pada Twitter dan mencapai akurasi hingga 92%, sementara CatBoost sebagai baseline mencapai 91,1%. Penelitian ini menegaskan bahwa model berbasis transformer mampu mengungguli pendekatan machine learning konvensional bahkan pada dataset monolingual berukuran sedang. Sejalan dengan itu, García et al. [15] mengevaluasi berbagai model transformer pada dataset berbahasa Inggris, meliputi TwHIN pada EXIST 2022 dengan F1-score 78,40%, DistilBETO pada HASOC 2021 dengan F1-score 86,56%, dan BERT pada EDOS 2023 dengan F1-score 73,80%. Hasil tersebut menunjukkan bahwa performa model transformer sangat dipengaruhi oleh karakteristik dan domain dataset yang digunakan, sehingga generalisasi lintas dataset tidak selalu dapat diasumsikan.

Pada konteks teks berbahasa Indonesia, beberapa penelitian telah mengeksplorasi pendekatan deep learning dan transformer untuk deteksi ujaran kebencian. Helmi et al. [18] menerapkan model LSTM dengan *embedding* GloVe pada data Twitter berbahasa Indonesia dan mencapai F1-score sebesar 94%, menunjukkan efektivitas representasi kata berbasis *embedding* dalam menangkap nuansa linguistik ujaran kebencian pada teks monolingual Indonesia. Marpaung et al. [19] mengombinasikan IndoBERT dengan arsitektur BiGRU dan

mencapai akurasi 84,77% pada dataset Twitter berbahasa Indonesia, membuktikan bahwa integrasi model bahasa berbasis transformer dengan arsitektur rekuren mampu meningkatkan kemampuan model dalam menyimpan informasi kontekstual dibandingkan metode konvensional seperti SVM. Sementara itu, Dharmawan et al. [20] mengembangkan model berbasis IndoBERT yang dikombinasikan dengan feedforward neural network (FFNN) untuk deteksi ujaran kebencian pada teks YouTube berbahasa Indonesia, dengan akurasi tertinggi 89,52%. Pendekatan ini menunjukkan bahwa model transformer berbasis bahasa Indonesia dapat diintegrasikan secara efektif ke dalam arsitektur klasifikasi yang lebih kompleks untuk meningkatkan performa deteksi.

Seiring meningkatnya penggunaan bahasa campuran di media sosial Indonesia, sejumlah penelitian mulai berfokus pada teks *code-mixed*. Arya Wicaksana et al. [21] membandingkan performa BLOOM-560M dan XLM-RoBERTa pada teks campuran Bahasa Indonesia dan Bahasa Inggris, dengan hasil menunjukkan bahwa BLOOM-560M mencapai akurasi 99,97% dan F1-score 99,97%, sementara XLM-RoBERTa mencapai akurasi 98,96% dan F1-score 98,69%. Namun, model dalam penelitian tersebut dilatih dan dievaluasi sepenuhnya menggunakan dataset yang dihasilkan melalui augmentasi berbasis GPT, sehingga distribusi data cenderung seragam dan tidak mencerminkan variasi linguistik informal yang sesungguhnya pada teks media sosial nyata. Hal ini menimbulkan pertanyaan mengenai kemampuan generalisasi model ketika dihadapkan pada teks dengan variasi ejaan, slang, dan ekspresi informal yang lazim ditemukan di media sosial. Pamungkas et al. [22] melakukan evaluasi mendalam terhadap berbagai pendekatan klasifikasi, meliputi Linear SVM, Decision Tree, Logistic Regression, Random Forest, LSTM, dan GRU, pada data media sosial berbahasa campuran Jawa–Indonesia dan Sunda–Indonesia. Hasil penelitian tersebut menunjukkan bahwa metode tradisional masih menghadapi keterbatasan yang signifikan dalam menangani teks *code-mixed* pada bahasa dengan sumber daya rendah, dengan F1-score terbaik berkisar antara 67% hingga 77% bergantung pada pasangan bahasa yang digunakan. Lebih lanjut, Pamungkas et al. [23] mengeksplorasi strategi augmentasi data berbasis parafrase GPT untuk meningkatkan performa deteksi ujaran kebencian pada teks campuran Indonesia–Jawa dan Indonesia–Sunda menggunakan XLM-RoBERTa, dengan F1-score masing-masing 76,8% dan 75,5%. Penelitian ini menegaskan potensi augmentasi data sintetis dalam mengatasi keterbatasan data teranotasi, meskipun performa yang dicapai masih berada di bawah model yang dilatih pada data yang lebih beragam dan representatif.

Penelitian pada teks *code-mixed* di luar konteks bahasa Indonesia juga telah menunjukkan perkembangan yang signifikan. Kathiravan et al. [24] mengusulkan metode Sentence Transfer Fine-tuning (SetFit) untuk mendeteksi konten ofensif pada teks campuran Tamil–Inggris, dengan capaian F1-score sebesar 88% dan performa yang melampaui model-model seperti mBERT dan IndicBERT. Aashna Jha et al. [25] mengembangkan pendekatan berbasis fitur MuRIL yang diklasifikasikan menggunakan TabNet untuk mendeteksi ujaran kebencian pada teks campuran Hindi–Inggris, baik dalam bentuk transliterasi maupun aksara Devanagari, dengan F1-score terbaik 89% menggunakan kombinasi TabNet dan MuRIL. Meskipun pendekatan-pendekatan ini efektif pada pasangan bahasa tertentu, keduanya belum secara eksplisit mengevaluasi ketahanan model terhadap variasi bahasa informal yang kompleks dan penggunaan slang yang lazim ditemukan dalam teks media sosial nyata.

Pada konteks bahasa selain Indonesia dan Inggris, beberapa penelitian juga memberikan kontribusi penting. Nauros Romim et al. [26] memperkenalkan HS-BAN, sebuah dataset benchmark berbahasa Bangla, dan menunjukkan bahwa *embedding* dari teks informal menghasilkan performa lebih baik dibandingkan teks formal, dengan Bi-LSTM mencapai F1-score 86,85%. Hashmi et al. [17] mengkaji deteksi ujaran kebencian pada Bahasa Norwegia menggunakan FastText, Bi-LSTM, GRU, serta berbagai model transformer multibahasa yang di-*fine-tune*, dan menerapkan LIME untuk meningkatkan transparansi sistem. Selain itu, García et al. [15] mengevaluasi beberapa model transformer pada dataset berbahasa Spanyol, dengan ALBETO mencapai F1-score 79,05%, DistilBETO 76,24%, dan MarIA 85,18%. Hasil tersebut menunjukkan bahwa performa model transformer sangat dipengaruhi oleh kesesuaian antara domain pelatihan dan domain evaluasi.

Berdasarkan tinjauan tersebut, meskipun *pre-trained multilingual transformer* menunjukkan performa unggul dalam deteksi ujaran kebencian, sebagian besar penelitian masih berfokus pada satu bahasa atau pasangan bahasa tertentu serta belum mengkaji secara mendalam pengaruh strategi *fine-tuning*, persiapan data, dan metode *ensemble* pada teks *code-mixed*. Selain itu, penerapan validasi silang berbasis *k-fold* pada deteksi ujaran kebencian multibahasa juga masih terbatas. Oleh karena itu, penelitian ini membandingkan model BLOOM-560M dan XLM-RoBERTa melalui proses *fine-tuning* serta mengevaluasi berbagai strategi *fine-tuning* dan metode *ensemble*, termasuk pendekatan 5-fold stacking, pada teks multilingual yang mencakup Bahasa Indonesia, Bahasa Inggris, dan *code-mixed*.

Tabel 2.1. Analisis komparatif untuk deteksi ujaran kebencian

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
Sukriti Narang et al. [14]	BERT	English	Twitter	92%	-	-	-	Mencapai akurasi 92% menggunakan BERT dan akurasi 91,1% menggunakan CatBoost dengan penyetelan hyperparameter untuk deteksi ujaran kebencian di Twitter.	-
	CatBoost			91,1%	-	-	-		
García et al. [15]	TwHIN	English	EXIST 2022	-	78,40%	75,69%	81,31%	Menunjukkan tingkat adaptabilitas dan efektivitas model transformer dalam meningkatkan kemampuan deteksi ujaran kebencian pada teks berbahasa Inggris.	-
	DistilBETO		HASOC 2021	-	86,56%	84,75%	88,47%		
	BERT		EDOS 2023	-	73,80%	71,92%	75,77%		
Helmi et al. [18]	GloVe+6 Layer	Indonesia	Twitter	94,24%	94,00%	89,00%	99%	Menunjukkan tingkat efektivitas yang tinggi dalam mengidentifikasi nuansa linguistik ujaran kebencian berbahasa Indonesia di Twitter, dengan nilai recall dan kinerja keseluruhan yang sangat memuaskan.	-
	LSTM+6 Layer			93,61%	93,00%	88,00%	99%		
	GloVe+5 Layer			94,24%	93,00%	88,00%	99%		
	LSTM+5 Layer			93,80%	93,00%	89,00%	99%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
Marpaung et al. [19]	BiGRU + IndoBERT	Indonesia	Twitter	84,77%	-	-	-	Menunjukkan bahwa kombinasi IndoBERT dan BiGRU dengan konfigurasi tanpa penghapusan *stop word* memiliki efektivitas terbaik dalam mendeteksi ujaran kebencian pada Twitter berbahasa Indonesia dengan akurasi mencapai 84,77%. Model berbasis deep learning ini terbukti lebih tangguh dibandingkan metode machine learning konvensional seperti SVM dan RFDT berkat kemampuannya yang unggul dalam menyimpan informasi kontekstual yang penting dari teks.	-
Dharmawan et al. [20]	IndoBERT + FFNN	Indonesia	Youtube	89,52%	-	-	-	Mengembangkan model klasifikasi berbasis feedforward neural network yang dikombinasikan dengan IndoBERT untuk mendeteksi ujaran kebencian pada teks berbahasa Indonesia. Model yang diintegrasikan ke dalam aplikasi berbasis web ini terbukti sangat efektif dan tangguh dalam membedakan ujaran kebencian dengan kalimat normal, serta berhasil mencapai tingkat akurasi tertinggi sebesar 89,52%.	-
Arya Wicaksana et al. [21]	BLOOM-560M	Indonesia-English	GPT Generated	99,97%	99,97%	100%	99,93%	Mencapai akurasi 99,97% dan F1-score 99,97% menggunakan BLOOM-560M pada dataset yang sepenuhnya dihasilkan melalui augmentasi berbasis GPT.	-
	XLM-RoBERTa			98,96%	98,69%	98,71%	98,69%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
Pamungkas et al. [22]	Linear SVM	Javanese	Social Media	84,9%	76,2%	78,9%	74,4%	Evaluasi mendalam terhadap model Transformer multibahasa untuk deteksi *hate speech* pada bahasa minim aset (low-resource), khususnya bahasa Indonesia, menunjukkan hasil yang signifikan. Pendekatan berbasis *pretrained multilingual models* (misalnya XLM-RoBERTa) terbukti secara konsisten melampaui performa model monolingual dan metode tradisional dalam hal akurasi serta kemampuan generalisasi.	-
	Linear SVM	Javanese-Indonesia	Social Media	79,8%	67,4%	70%	66%		
	Linear SVM	Javanese-English	Social Media	80,4%	67,3%	71,2%	65,5%		
	Linear SVM	Sundanese	Social Media	79,4%	70,8%	73%	69,5%		
	Linear SVM	Sundanese-Indonesia	Social Media	77%	67,9%	69,4%	67%		
	Linear SVM	Sundanese-English	Social Media	78,7%	69,4%	72%	68%		
	Decision Tree	Javanese	Social Media	83,9%	75,1%	76,8%	73,9%		
	Decision Tree	Javanese-Indonesia	Social Media	77,6%	63,4%	65,8%	62,2%		
	Decision Tree	Javanese-English	Social Media	77,2%	62,9%	65,1%	61,8%		
	Decision Tree	Sundanese	Social Media	77,1%	68,1%	69,6%	67,2%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
	Decision Tree	Sundanese-Indonesia	Social Media	74,7%	65%	66%	64,4%		
	Decision Tree	Sundanese-English	Social Media	75%	67,8%	67,5%	68,2%		
	Logistic Regression	Javanese	Social Media	86,3%	76,9%	83%	73,8%		
	Logistic Regression	Javanese-Indonesia	Social Media	81,7%	67,5%	74,8%	65%		
	Logistic Regression	Javanese-English	Social Media	82%	65,4%	77,8%	63%		
	Logistic Regression	Sundanese	Social Media	81,1%	71,7%	76,6%	69,6%		
	Logistic Regression	Sundanese-Indonesia	Social Media	78,1%	65,6%	71,7%	64%		
	Logistic Regression	Sundanese-English	Social Media	79,2%	66,5%	74,6%	64,5%		
	Random Forest	Javanese	Social Media	84,1%	70,4%	82,6%	67%		
	Random Forest	Javanese-Indonesia	Social Media	80,6%	58,8%	77,6%	58,1%		
	Random Forest	Javanese-English	Social Media	79%	59,2%	68,3%	58,3%		
	Random Forest	Sundanese	Social Media	80,1%	68%	76,6%	65,7%		
	Random Forest	Sundanese-Indonesia	Social Media	79,2%	65,1%	76%	63,2%		
	Random Forest	Sundanese-English	Social Media	78,7%	68,4%	72,3%	66,7%		
	LSTM	Javanese	Social Media	85,1%	73,9%	81,9%	70,6%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
	LSTM	Javanese-Indonesia	Social Media	80,8%	61,9%	74,8%	60,2%		
	LSTM	Javanese-English	Social Media	80,8%	66,8%	72,3%	64,7%		
	LSTM	Sundanese	Social Media	78,6%	72,7%	72,1%	73,3%		
	LSTM	Sundanese-Indonesia	Social Media	77,1%	69,8%	69,9%	69,7%		
	LSTM	Sundanese-English	Social Media	75,6%	69,5%	68,7%	70,5%		
	GRU	Javanese	Social Media	84,9%	73,2%	82,3%	69,7%		
	GRU	Javanese-Indonesia	Social Media	81,3%	62,8%	76,6%	60,9%		
	GRU	Javanese-English	Social Media	81,1%	64,1%	74,5%	62%		
	GRU	Sundanese	Social Media	79,4%	72,2%	72,9%	71,6%		
	GRU	Sundanese-Indonesia	Social Media	75,9%	67,6%	68,1%	67,3%		
	GRU	Sundanese-English	Social Media	72,1%	67,3%	66,6%	70,2%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
Pamungkas et al. [23]	XLM-RoBERTa	Indonesia-Javanese	GPT Augmented	-	76,8%	-	-	Menunjukkan bahwa strategi augmentasi data berbasis parafrase GPT terbukti efektif dalam meningkatkan performa model deteksi ujaran kebencian pada teks Twitter berbahasa campuran (*code-mixed*) Indonesia. Pendekatan ini menegaskan ketangguhan model berbasis transformer, khususnya IndoBERT dan XLM-RoBERTa, dalam mengatasi keterbatasan data teranotasi (low-resource) serta variabilitas linguistik pada media sosial dengan hasil yang lebih kuat dibandingkan model tradisional dan RNN.	-
	XLM-RoBERTa	Indonesia-Sundanese	GPT Augmented	-	75,5%	-	-		-
Kathiravan et al. [24]	SetFit	Tamil-English	Social Media	89,00%	88,00%	90,00%	87,00%	Menunjukkan bahwa SetFit memiliki efektivitas yang lebih baik dibandingkan model tradisional, serta menegaskan ketangguhannya dalam menangani tantangan deteksi konten ofensif pada teks berbahasa campuran dengan tingkat akurasi tinggi.	-
	mBERT			86,00%	84,00%	88,00%	84,00%		
	LSTM			76,00%	67,00%	70,00%	65,00%		
	BERT			78,00%	70,00%	72,00%	68,00%		
	indicBERT			80,00%	72,00%	74,00%	70,00%		
	LaBSE			84,00%	79,00%	80,00%	78,00%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
Aashna Jha et al. [25]	TabNet+PCA+M-BERT	Hindi-English	Twitter	65%	65%	64%	66%	Mengembangkan model klasifikasi berbasis TabNet yang dilatih menggunakan fitur hasil ekstraksi dari MuRIL untuk teks campuran Hindi–Inggris. Model ini terbukti dapat bekerja baik pada data *code-mixed* yang ditransliterasi maupun teks dengan aksara Devanagari asli.	-
	TabNet+PCA+Indic-BERT			71%	71%	72%	71%		
	TabNet+PCA+XLM-RoBERTa			84%	85%	84%	84%		
	TabNet+PCA+MuRIL			90%	89%	89%	89%		
Nauros Romim et al. [26]	Bi-LSTM+FT(SG)	Bangla	Facebook, Youtube	-	86,85%	-	-	Mencapai F1-score sebesar 86,85% dengan model Bi-LSTM yang dibangun di atas *word embedding* FastText informal untuk deteksi ujaran kebencian berbahasa Bangla. *Embedding* teks informal memberikan performa lebih baik dibandingkan *embedding* teks formal.	-
	Bangla-BERT			-	83,68%	-	-		
	CNN+BFT			-	71,58%	-	-		
Hashmi et al. [17]	mBERT	Norwegia	Resett, Twitter, Facebook	82,00%	79,00%	78,00%	82,00%	Mengintegrasikan arsitektur jaringan saraf tingkat lanjut dengan teknik interpretabilitas untuk meningkatkan kinerja sekaligus transparansi sistem deteksi ujaran kebencian pada skenario data terbatas.	LIME
	ELECTRAbase			83,00%	75,00%	69,00%	83,00%		
	ELECTRAlarge			83,00%	75,00%	69,00%	83,00%		

Lanjut pada halaman berikutnya

Tabel 2.1 Analisis komparatif untuk deteksi ujaran kebencian (Lanjutan)

Studi	Model	Bahasa	Dataset	Akurasi	F1-score	Presisi	Recall	State Of The Art (SOTA)	XAI
	scandi-BERT			81,00%	80,00%	79,00%	81,00%		
	nb-BERTbase			81,00%	81,00%	81,00%	81,00%		
	nb-BERTlarge			81,00%	81,00%	81,00%	81,00%		
	Nor-BERTsmall			82,00%	79,00%	77,00%	83,00%		
	Nor-BERTbase			82,00%	80,00%	78,00%	83,00%		
	Nor-BERTlarge			83,00%	81,00%	80,00%	82,00%		
	FLAN-T5-small			80,00%	77,00%	76,00%	80,00%		
	FLAN-T5-base			83,00%	80,00%	82,00%	82,00%		
	Nor-T5small			77,00%	78,00%	77,00%	78,00%		
García et al. [15]	ALBETO	Spanish	EXIST 2022	-	79,05%	77,18%	75,39%	Menunjukkan tingkat adaptabilitas dan efektivitas model transformer dalam meningkatkan kemampuan deteksi ujaran kebencian pada teks berbahasa Spanyol.	-
	DistilBETO		HatEval 2019	-	76,24%	70,58%	82,88%		
	MarIA		Football 2023	-	85,18%	87,53%	82,94%		

2.2 Word Embedding

Word embedding merupakan teknik representasi kata ke dalam bentuk vektor bilangan real berdimensi tetap (*fixed-dimensional*), seperti 768 dimensi pada XLM-RoBERTa dan 1024 dimensi pada BLOOM-560M, yang mampu menangkap hubungan semantik dan sintaksis antar kata [27]. Berbeda dengan representasi *one-hot encoding* yang bersifat sparse dan dimensinya selalu sama dengan ukuran kosakata $|V|$ (dengan hanya satu elemen bernilai 1) sehingga ikut membesar seiring bertambahnya kosakata, *word embedding* memetakan setiap kata ke dalam ruang vektor kontinu yang dimensinya, d, ditentukan secara tetap sebagai *hyperparameter* sejak awal dan tidak bergantung pada $|V|$, di mana kata-kata dengan makna atau konteks penggunaan yang serupa akan memiliki representasi vektor yang berdekatan secara geometris. Karena d tidak bergantung pada $|V|$ dan dalam praktiknya dipilih jauh lebih kecil dibandingkan $|V|$ ($d \ll |V|$), representasi *word embedding* bersifat lebih ringkas (*dense*) dibandingkan *one-hot encoding*, meskipun nilai d itu sendiri — seperti 768 atau 1024 pada model yang digunakan dalam penelitian ini — dapat tergolong besar secara absolut.

Secara formal, *word embedding* dapat didefinisikan sebagai fungsi pemetaan $E : V \rightarrow \mathbb{R}^d$, di mana V adalah kosakata (*vocabulary*) dan d adalah dimensi ruang *embedding* yang jauh lebih kecil dibandingkan ukuran kosakata $|V|$. Pemetaan ini umumnya direpresentasikan sebagai matriks *embedding* $\mathbf{E} \in \mathbb{R}^{|V| \times d}$, di mana setiap baris matriks merepresentasikan vektor *embedding* untuk satu kata dalam kosakata.

Salah satu pendekatan populer untuk mempelajari *word embedding* adalah Word2Vec yang diperkenalkan oleh Mikolov et al. [27], yang terdiri dari dua arsitektur utama, yaitu *Continuous Bag-of-Words* (CBOW) dan *Skip-gram*. CBOW memprediksi kata target berdasarkan kata-kata konteks di sekitarnya, sedangkan *Skip-gram* bekerja sebaliknya, yaitu memprediksi kata-kata konteks berdasarkan satu kata target. Pendekatan lain yang juga banyak digunakan adalah GloVe (*Global Vectors for Word Representation*) yang diperkenalkan oleh Pennington et al. [28], yang mempelajari representasi kata berdasarkan statistik ko-okurensi (*co-occurrence*) kata pada seluruh korpus, sehingga menggabungkan keunggulan metode berbasis matriks faktorisasi global dengan metode *local context window*.

Kedua pendekatan tersebut menghasilkan *embedding* yang bersifat statis (*static embedding*), yaitu setiap kata memiliki tepat satu representasi vektor yang tetap terlepas dari konteks kalimat tempat kata tersebut muncul, sehingga tidak mampu menangani fenomena polisemi (satu kata dengan makna berbeda

bergantung konteks). Keterbatasan ini menjadi salah satu motivasi utama berkembangnya *contextual embedding*, yaitu representasi kata yang dihasilkan oleh model bahasa berbasis *deep learning* seperti Transformer, di mana representasi vektor suatu kata dapat berubah bergantung pada konteks kalimat di sekitarnya [29]. Model seperti BERT dan XLM-RoBERTa yang digunakan dalam penelitian ini menghasilkan *contextual embedding* melalui mekanisme *self-attention*, sehingga representasi yang dihasilkan jauh lebih kaya dan sensitif terhadap konteks dibandingkan *word embedding* statis seperti Word2Vec maupun GloVe.

2.3 Term Frequency-Inverse Document Frequency (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) merupakan teknik pembobotan statistik yang digunakan untuk merepresentasikan pentingnya suatu kata (*term*) terhadap sebuah dokumen dalam kumpulan dokumen (*corpus*) [30]. Berbeda dengan *word embedding* yang menghasilkan representasi vektor berdimensi rendah dan padat (*dense*), TF-IDF menghasilkan representasi vektor berdimensi tinggi dan jarang (*sparse*) sebesar ukuran kosakata, di mana setiap elemen vektor merepresentasikan skor kepentingan suatu kata dalam dokumen tertentu.

TF-IDF terdiri atas dua komponen utama. Komponen pertama, *Term Frequency* (TF), mengukur frekuensi kemunculan suatu kata t dalam sebuah dokumen d:

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.1}$$

di mana $f_{t,d}$ adalah jumlah kemunculan kata t dalam dokumen d, dan penyebut merupakan jumlah total kemunculan seluruh kata dalam dokumen tersebut, sehingga TF merepresentasikan proporsi relatif kemunculan suatu kata terhadap panjang dokumen.

Komponen kedua, *Inverse Document Frequency* (IDF), mengukur seberapa jarang suatu kata muncul di seluruh dokumen dalam korpus, sehingga memberikan bobot yang lebih rendah pada kata-kata umum (misalnya kata sambung) yang muncul di hampir seluruh dokumen dan bobot yang lebih tinggi pada kata-kata yang lebih jarang namun lebih diskriminatif:

$$\text{IDF}(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right) \tag{2.2}$$

di mana N adalah jumlah total dokumen dalam korpus D, dan $|\{d \in D : t \in d\}|$ adalah jumlah dokumen yang memuat kata t minimal satu kali.

Skor akhir TF-IDF untuk suatu kata dalam suatu dokumen diperoleh melalui perkalian kedua komponen tersebut:

$$\text{TF-IDF}(t,d,D) = \text{TF}(t,d) \times \text{IDF}(t,D) \tag{2.3}$$

Skor TF-IDF yang tinggi menunjukkan bahwa suatu kata sering muncul dalam sebuah dokumen tertentu namun jarang muncul pada dokumen-dokumen lain dalam korpus, sehingga kata tersebut dianggap memiliki daya pembeda yang tinggi untuk merepresentasikan konten dokumen tersebut secara spesifik [30]. Representasi TF-IDF umumnya digunakan sebagai fitur masukan bagi algoritma *machine learning* konvensional seperti SVM, Logistic Regression, dan Random Forest, sebagaimana yang diterapkan pada sejumlah penelitian deteksi ujaran kebencian terdahulu [22]. Meskipun demikian, TF-IDF memiliki keterbatasan karena tidak mampu menangkap hubungan semantik maupun urutan kata dalam kalimat, berbeda dengan *word embedding* maupun representasi kontekstual berbasis Transformer.

2.4 Arsitektur Transformer

Transformer merupakan arsitektur jaringan saraf yang diperkenalkan oleh Vaswani et al. [31] sebagai alternatif terhadap model sekuensial berbasis recurrent neural network (RNN) dan long short-term memory (LSTM). Keunggulan utama transformer terletak pada mekanisme *self-attention* yang memungkinkan model memproses seluruh token dalam sebuah urutan secara paralel, sehingga lebih efisien secara komputasi dan lebih mampu menangkap ketergantungan jangka panjang antar token [31].

2.4.1 Positional Encoding dan Alur Kerja Transformer

Karena mekanisme *self-attention* memproses seluruh token secara paralel tanpa urutan yang melekat, Transformer memerlukan mekanisme tambahan untuk menyisipkan informasi posisi setiap token dalam urutan, yang disebut *positional encoding* [31]. Pada arsitektur Transformer asli, *positional encoding* dihasilkan menggunakan fungsi sinusoidal dengan frekuensi yang berbeda-beda untuk setiap dimensi:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), \quad PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.4}$$

di mana *pos* adalah posisi token dalam urutan, i adalah indeks dimensi, dan d_{model} adalah dimensi *embedding*. Vektor *positional encoding* kemudian dijumlahkan dengan vektor *word embedding* sebelum diumpankan ke lapisan *encoder* pertama.

Secara keseluruhan, alur kerja Transformer dimulai dari lapisan *embedding* yang mengubah setiap token menjadi vektor representasi, dijumlahkan dengan *positional encoding*, kemudian diproses secara berurutan melalui tumpukan (*stack*) blok *encoder* atau *decoder*. Setiap blok terdiri atas lapisan *multi-head self-attention* yang diikuti oleh lapisan *feed-forward* (*Multi-Layer Perceptron*), dengan koneksi residual (*residual connection*) dan normalisasi lapisan (*layer normalization*) diterapkan setelah masing-masing sub-lapisan untuk menjaga stabilitas gradien selama pelatihan [31]. Output dari blok terakhir kemudian digunakan sebagai representasi akhir, baik untuk tugas klasifikasi (melalui token `[CLS]` pada model *encoder-only*) maupun untuk prediksi token berikutnya secara autoregresif (pada model *decoder-only* seperti BLOOM). Sebagai catatan, model berbasis ALiBi seperti BLOOM menggantikan *positional encoding* sinusoidal tersebut dengan bias linear yang ditambahkan langsung pada skor *attention*, sebagaimana telah dijelaskan pada persamaan (2.7).

2.4.2 Mekanisme Self-Attention

Mekanisme *self-attention* menghitung representasi setiap token berdasarkan hubungannya dengan seluruh token lain dalam urutan yang sama [31]. Secara matematis, *self-attention* dihitung menggunakan tiga matriks yang diproyeksikan dari representasi input, yaitu query (Q), key (K), dan value (V), dengan formula

sebagai berikut:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.5}$$

di mana d_k merupakan dimensi vektor key yang digunakan sebagai faktor skala untuk menjaga kestabilan gradien selama pelatihan [31]. Fungsi softmax mengubah skor perhatian menjadi distribusi probabilitas, sehingga setiap token memperoleh bobot kontribusi terhadap representasi token yang sedang dihitung.

2.4.3 Multi-Head Attention

Transformer menggunakan *multi-head attention*, yaitu mekanisme yang menjalankan *self-attention* secara paralel pada beberapa subspace representasi yang berbeda [31]. Hal ini memungkinkan model untuk secara bersamaan menangkap berbagai jenis hubungan antar token, seperti hubungan sintaksis dan semantik, dari perspektif yang berbeda-beda. Output dari setiap head digabungkan dan diproyeksikan kembali menggunakan matriks bobot:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O \tag{2.6}$$

di mana setiap $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ dan W^O merupakan matriks proyeksi output [31].

2.4.4 Encoder dan Decoder

Arsitektur transformer asli terdiri atas dua komponen utama, yaitu *encoder* dan *decoder* [31]. *Encoder* memproses seluruh urutan input secara bidireksional dan menghasilkan representasi kontekstual untuk setiap token. *Decoder* menggunakan representasi dari *encoder* bersama dengan masked *self-attention* untuk menghasilkan urutan output secara autoregressif. Dalam konteks model bahasa modern, beberapa arsitektur hanya menggunakan *encoder* (seperti BERT [29] dan XLM-RoBERTa [32]) untuk tugas pemahaman teks seperti klasifikasi, sementara arsitektur *decoder-only* (seperti BLOOM [33]) digunakan untuk tugas generasi teks.

2.5 Model BLOOM

BigScience Large Open-science Open-access Multilingual Language Model (BLOOM) merupakan model bahasa multibahasa open-access dengan 176 miliar parameter yang dirancang untuk melakukan berbagai tugas pemrosesan dalam berbagai bahasa. Model ini dikembangkan melalui kolaborasi berskala besar yang melibatkan ratusan peneliti, dan dibangun menggunakan arsitektur Transformer *decoder-only*. BLOOM dilatih menggunakan korpus ROOTS, yaitu kumpulan data berskala besar yang dihimpun dari ratusan sumber dalam 46 bahasa alami serta 13 bahasa pemrograman. Berbeda dengan sebagian besar model bahasa berukuran besar yang umumnya dikembangkan oleh organisasi dengan sumber daya komputasi tinggi dan bersifat tertutup, BLOOM dirancang agar dapat diakses secara publik serta menunjukkan performa yang kompetitif pada berbagai tolok ukur, dengan capaian yang sangat baik terutama setelah melalui *multitask prompted fine-tuning*. Proses pelatihan BLOOM dilakukan menggunakan superkomputer Jean Zay yang didanai oleh pemerintah Prancis, yang dimiliki oleh Grand équipement national de calcul intensif (GENCI), dan dioperasikan di Institut du développement et des ressources en informatique scientifique (IDRIS) sebagai pusat komputasi nasional milik French National Center for Scientific Research (CNRS) [33].

BLOOM dibangun berdasarkan arsitektur transformer *decoder-only* dengan 70 lapisan dan 176 miliar parameter, yang dirancang khusus untuk memproses dan menghasilkan teks secara berurutan. Model ini menggunakan mekanisme *self-attention* yang memungkinkan BLOOM untuk memahami hubungan dan konteks antar kata dalam sebuah kalimat, serta dilengkapi dengan inovasi ALiBi (Attention with Linear Biases) untuk meningkatkan kinerja dan stabilitas pelatihan. ALiBi merupakan metode pengkodean posisi yang digunakan pada BLOOM sebagai alternatif terhadap *positional embedding* sinusoidal maupun learned *positional embedding* yang umum digunakan pada model transformer sebelumnya [34]. Berbeda dengan pendekatan konvensional yang menambahkan vektor posisi pada representasi token, ALiBi secara langsung menambahkan bias linear pada skor attention sebelum operasi softmax dilakukan:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + m \cdot B\right)V \tag{2.7}$$

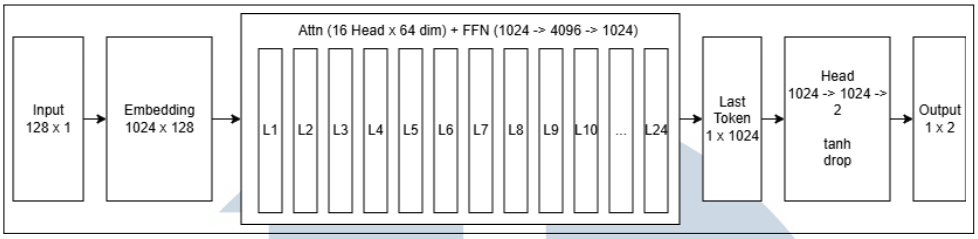
di mana m adalah kemiringan (*slope*) yang bersifat spesifik untuk setiap *attention*

head dan B adalah matriks bias yang nilainya proporsional terhadap jarak antara posisi *query* dan *key*. Semakin jauh jarak antar token, semakin besar nilai bias negatif yang diberikan, sehingga model secara implisit memberikan bobot perhatian yang lebih tinggi pada token-token yang berdekatan. Pendekatan ini memungkinkan BLOOM untuk melakukan ekstrapolasi pada urutan yang lebih panjang dari yang dilihat selama pelatihan tanpa memerlukan modifikasi arsitektur tambahan [34].

Sebagai model *decoder-only* yang dirancang untuk pemodelan bahasa secara autoregressif, BLOOM tidak memiliki mekanisme bawaan untuk tugas klasifikasi urutan teks. Adaptasi BLOOM untuk tugas klasifikasi dilakukan dengan mengekstraksi *hidden state* dari token terakhir yang bukan merupakan token *padding* sebagai representasi keseluruhan urutan teks, kemudian meneruskannya ke lapisan klasifikasi tambahan. Pendekatan ini berbeda dengan model berbasis *encoder* seperti XLM-RoBERTa yang menggunakan token `[CLS]` di awal urutan sebagai representasi kalimat.

Karena BLOOM memproses token secara kausal dari kiri ke kanan, penggunaan token terakhir sebagai representasi kalimat mensyaratkan penerapan *left padding*, yaitu penambahan token kosong di sisi kiri urutan. Dengan *left padding*, token terakhir yang diproses model selalu merupakan token bermakna dari teks, bukan token *padding*, sehingga representasi yang diekstraksi mencerminkan konteks keseluruhan teks secara akurat. Representasi tersebut kemudian diproses melalui lapisan klasifikasi yang terdiri dari proyeksi $1024 \rightarrow 1024$ dengan fungsi aktivasi *tanh*, *dropout* sebesar 0.1, dan lapisan *output* $1024 \rightarrow 2$ untuk menghasilkan prediksi kelas akhir.

BLOOM dilatih menggunakan korpus ROOTS yang terdiri dari 1.61 terabyte data teks multibahasa dari 46 bahasa alami dan 13 bahasa pemrograman, yang bersumber dari berbagai media seperti buku, artikel ilmiah, dan situs web. Model ini memproses teks dengan mengubahnya menjadi token, satuan-satuan kecil seperti kata atau sub-kata dan menggunakan tokenizer *byte-level BPE* dengan kosakata sebanyak 250,680 token. Proses ini dimulai dari lapisan *embedding*, kemudian melalui *decoder* blocks yang terdiri dari *multi-head attention* dan feed-forward layers (MLP) seperti yang ditunjukkan pada Gambar 2.1.



Gambar 2.1. Arsitektur BLOOM

Melalui pendekatan *causal language modeling*, BLOOM dapat memprediksi token berikutnya secara autoregressif, yang memungkinkannya menghasilkan teks yang koheren dan kontekstual. Kemampuan ini menjadikan BLOOM dapat melakukan berbagai tugas pemrosesan bahasa alami seperti terjemahan, peringkasan, menjawab pertanyaan, dan bahkan menghasilkan kode program dalam berbagai bahasa [33].

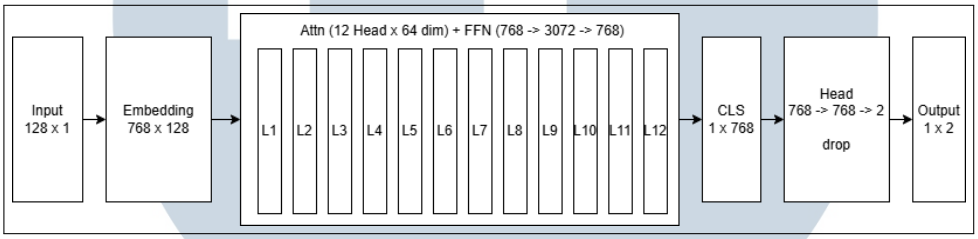
Dalam penelitian ini digunakan BLOOM-560M, yaitu varian BLOOM dengan skala yang lebih kecil (560 juta parameter) dibandingkan model BLOOM versi penuh yang berukuran sangat besar. Pemilihan varian ini didorong oleh pertimbangan keterjangkauan komputasi, sehingga proses pelatihan, pengujian, dan iterasi eksperimen dapat dilakukan secara efisien dan realistis dalam keterbatasan sumber daya penelitian.

Walaupun kapasitas parameternya lebih rendah, BLOOM-560M tetap mempertahankan rancangan inti arsitektur Transformer yang menjadi fondasi BLOOM, sehingga tetap relevan untuk memodelkan konteks pada data multibahasa dan teks *code-mixed* yang dianalisis. Ukuran yang lebih ringkas juga memudahkan pemanfaatan lingkungan komputasi yang umum digunakan dalam penelitian, seperti Google Colab Pro, untuk menjalankan proses *fine-tuning* dan evaluasi tanpa ketergantungan pada infrastruktur komputasi berskala besar.

2.6 Model XLM-RoBERTa

Cross-lingual Language Model - Robustly Optimized BERT Pretraining Approach (XLM-RoBERTa) merupakan model bahasa multibahasa berbasis arsitektur Transformer *encoder* yang dikembangkan oleh Conneau et al. [32] dari Facebook AI Research. Model ini merupakan pengembangan dari RoBERTa yang dilatih secara khusus pada korpus multibahasa berskala besar, sehingga mampu memahami dan memproses teks dalam lebih dari 100 bahasa secara terpadu dalam satu representasi bersama.

XLM-RoBERTa dibangun berdasarkan arsitektur Transformer *encoder-only* dengan mekanisme *self-attention* bidireksional, yang memungkinkan model untuk memahami konteks sebuah token berdasarkan seluruh token di sekitarnya, baik dari sisi kiri maupun kanan dalam satu urutan teks. Berbeda dengan BLOOM yang menggunakan arsitektur *decoder-only* untuk pemodelan bahasa secara autoregressif, XLM-RoBERTa dirancang khusus untuk tugas-tugas pemahaman teks seperti klasifikasi, sehingga lebih sesuai untuk pendeteksian ujaran kebencian yang pada dasarnya merupakan tugas klasifikasi urutan teks seperti yang ditunjukkan pada Gambar 2.2.



Gambar 2.2. Arsitektur XLM-RoBERTa

Model ini dilatih menggunakan pendekatan *Masked Language Modeling (MLM)* pada korpus CommonCrawl yang terdiri dari 2,5 terabyte data teks multibahasa, mencakup lebih dari 100 bahasa termasuk Bahasa Indonesia dan Bahasa Inggris. Secara formal, objektif MLM meminimalkan *negative log-likelihood* pada posisi-posisi token yang dimasking:

$$\mathscr{L}_{\text{MLM}} = -\sum_{i \in \mathscr{M}} \log P(x_i \mid x_{\setminus \mathscr{M}}) \tag{2.8}$$

di mana $\mathscr{M}$ adalah himpunan posisi token yang di-*mask*, x_i adalah token asli pada posisi tersebut, dan $x_{\setminus \mathscr{M}}$ adalah seluruh token dalam urutan kecuali token yang di-*mask*. Berbeda dengan BERT asli yang menerapkan *masking* satu kali selama pra-pemrosesan, XLM-RoBERTa menggunakan *dynamic masking* di mana pola *mask* dibangkitkan ulang setiap kali urutan yang sama diumpankan ke model selama pelatihan. Pendekatan ini meningkatkan keragaman data pelatihan secara efektif tanpa memerlukan penambahan data baru [32].

Untuk tugas klasifikasi teks, XLM-RoBERTa memanfaatkan token khusus `[CLS]` yang ditempatkan di awal setiap urutan masukan. Selama proses *fine-tuning*, *hidden state* dari token `[CLS]` pada lapisan terakhir, $\mathbf{h}_{\texttt{[CLS]}} \in \mathbb{R}^{768}$, digunakan sebagai representasi kontekstual dari keseluruhan urutan teks. Representasi ini

mengandung informasi bidireksional dari seluruh token dalam urutan karena mekanisme *self-attention* pada model berbasis *encoder* memungkinkan setiap token, termasuk `[CLS]`, untuk memperhatikan seluruh token lain dari kedua arah secara bersamaan. $\mathbf{h}_{[\texttt{CLS}]}$ kemudian diproses melalui lapisan klasifikasi yang terdiri dari proyeksi $768 \rightarrow 768$ dengan fungsi aktivasi *tanh*, *dropout* sebesar 0.1, dan lapisan *output* $768 \rightarrow 2$ untuk menghasilkan prediksi kelas akhir [32].

Proses pelatihan yang lebih robust dibandingkan model BERT asli, yang meliputi penghapusan *next sentence prediction*, serta pelatihan dengan batch size yang lebih besar dan durasi yang lebih panjang, menjadikan XLM-RoBERTa lebih stabil dan konsisten dalam menghasilkan representasi teks yang berkualitas tinggi.

Selain itu, XLM-RoBERTa menggunakan tokenizer SentencePiece dengan model *unigram* dan kosakata sebanyak 250.002 token yang mencakup lebih dari 100 bahasa secara terpadu [32]. Kosakata bersama ini memungkinkan *subword* dari bahasa Indonesia dan bahasa Inggris untuk hidup berdampingan dalam ruang *embedding* yang sama, sehingga model dapat menangani teks *code-mixed* tanpa memerlukan pemisahan atau deteksi bahasa secara eksplisit terlebih dahulu.

Meskipun demikian, pelatihan pada korpus multibahasa yang sangat besar menimbulkan fenomena yang dikenal sebagai *curse of multilinguality* [32], yaitu kondisi di mana peningkatan jumlah bahasa yang didukung menyebabkan penurunan kapasitas representasi per bahasa akibat persaingan ruang parameter yang tetap. Fenomena ini berdampak pada penurunan performa model pada bahasa-bahasa tertentu, khususnya bahasa dengan representasi data pelatihan yang lebih rendah, dibandingkan dengan model monolingual yang dilatih secara khusus pada satu bahasa. Hal ini relevan dalam konteks penelitian ini karena dapat menjelaskan mengapa performa model pada teks berbahasa Inggris lebih rendah dibandingkan model monolingual yang dilaporkan dalam penelitian terdahulu.

Dalam penelitian ini digunakan varian XLM-RoBERTa-base yang memiliki 12 lapisan *encoder*, 768 dimensi hidden state, 12 attention heads, dan total 270 juta parameter. Pemilihan varian base didorong oleh pertimbangan keseimbangan antara kapasitas model dan efisiensi komputasi, sehingga proses *fine-tuning* dapat dilakukan secara realistis dalam lingkungan Google Colab Pro. Kemampuan XLM-RoBERTa dalam memproses teks multibahasa secara terpadu menjadikannya kandidat yang relevan untuk menangani teks *code-mixed* Bahasa Indonesia–Inggris yang menjadi fokus penelitian ini.

2.7 Fine-tuning Model Transformer

Fine-tuning merupakan proses adaptasi model bahasa yang telah dilatih secara umum (*pre-trained*) terhadap tugas atau domain spesifik menggunakan dataset berlabel yang lebih kecil [29]. Pendekatan ini didasarkan pada prinsip transfer learning, di mana pengetahuan linguistik yang diperoleh selama pre-training pada korpus berskala besar dapat dipindahkan dan disesuaikan untuk tugas hilir (downstream task) tertentu [29].

Dalam proses *fine-tuning* untuk klasifikasi teks, sebuah lapisan klasifikasi ditambahkan di atas representasi output model *pre-trained*, kemudian seluruh parameter diperbarui menggunakan data berlabel melalui minimisasi fungsi loss [29]. Fungsi loss yang umum digunakan untuk klasifikasi biner adalah *cross-entropy loss*:

$$\mathscr{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)] \tag{2.9}$$

di mana y_i adalah label sebenarnya, $\hat{y}_i$ adalah probabilitas prediksi, dan N adalah jumlah sampel. Dalam konteks data yang tidak seimbang, fungsi loss dapat dimodifikasi menggunakan *class weights* untuk memberikan penalti yang lebih besar terhadap kesalahan prediksi pada kelas minoritas [35, 36].

Terdapat beberapa strategi *fine-tuning* yang umum dieksplorasi dalam penelitian NLP [37–39]:

1. **Full fine-tuning**: Seluruh parameter model diperbarui selama pelatihan. Pendekatan ini umumnya menghasilkan performa terbaik namun memerlukan sumber daya komputasi yang lebih besar.

2. **Layer freezing**: Sejumlah lapisan awal model dibekukan (tidak diperbarui) selama pelatihan, sehingga hanya lapisan atas yang beradaptasi terhadap tugas baru. Pendekatan ini mengurangi kebutuhan komputasi dan dapat mengurangi risiko overfitting pada dataset kecil.

3. **Layer-wise Learning Rate Decay (LLRD)**: Setiap lapisan diberikan learning rate yang berbeda, di mana lapisan yang lebih dekat dengan input menggunakan learning rate yang lebih kecil dibandingkan lapisan yang lebih dekat dengan output. Pendekatan ini bertujuan untuk menjaga representasi

umum yang telah dipelajari selama pre-training sambil tetap mengadaptasi lapisan atas terhadap tugas spesifik.

4. **Custom classification head**: Lapisan klasifikasi yang lebih kompleks ditambahkan di atas *encoder*, misalnya dengan menambahkan lapisan tersembunyi tambahan, fungsi aktivasi, dan dropout, untuk meningkatkan kapasitas representasi sebelum prediksi akhir.

2.8 Metode Ensemble

Metode *ensemble* merupakan pendekatan yang menggabungkan prediksi dari beberapa model untuk menghasilkan prediksi akhir yang lebih akurat dan robust dibandingkan model tunggal [40]. Pendekatan ini bekerja dengan cara memanfaatkan keberagaman dari berbagai model dasar, di mana kelemahan atau kesalahan prediksi dari satu model dapat dikoreksi oleh model lainnya

2.8.1 Soft Voting

Soft voting menggabungkan probabilitas prediksi dari setiap model dan mengambil rata-rata sebagai prediksi akhir [41]. Untuk M model dengan probabilitas prediksi $p_m(y \mid x)$ untuk kelas y dan input x, prediksi akhir dihitung sebagai [42]:

$$\hat{y} = \arg\max_y \frac{1}{M} \sum_{m=1}^{M} p_m(y \mid x) \tag{2.10}$$

2.8.2 Weighted Soft Voting

Weighted soft voting memberikan bobot yang berbeda pada setiap model berdasarkan performanya, sehingga model dengan performa lebih baik memiliki pengaruh yang lebih besar terhadap prediksi akhir [42, 43]:

$$\hat{y} = \arg\max_y \sum_{m=1}^{M} w_m \cdot p_m(y \mid x), \quad \sum_{m=1}^{M} w_m = 1 \tag{2.11}$$

di mana w_m adalah bobot untuk model ke-m yang ditentukan melalui pencarian grid (*grid search*) pada data validasi.

2.8.3 Stacking dengan Meta-learner

Stacking merupakan teknik ensemble di mana prediksi dari beberapa model tingkat pertama (base models) digunakan sebagai fitur masukan untuk melatih model tingkat kedua yang disebut *meta-learner* [44–46]. *Meta-learner* belajar bagaimana mengombinasikan prediksi base models secara optimal berdasarkan data validasi. Pendekatan ini lebih fleksibel dibandingkan voting karena *meta-learner* dapat mempelajari pola kesalahan dari setiap base model dan mengkompensasinya.

2.9 Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan algoritma *supervised learning* yang bekerja dengan mencari *hyperplane* pemisah optimal yang memaksimalkan margin antara dua kelas data [47]. Margin didefinisikan sebagai jarak antara *hyperplane* pemisah dengan titik data terdekat dari masing-masing kelas, yang disebut *support vectors*. Untuk data yang dapat dipisahkan secara linear, *hyperplane* pemisah dirumuskan sebagai:

$$f(x) = \mathbf{w}^T x + b \tag{2.12}$$

di mana $\mathbf{w}$ adalah vektor bobot yang tegak lurus terhadap *hyperplane*, dan b adalah bias. Proses pelatihan SVM bertujuan menemukan $\mathbf{w}$ dan b yang memaksimalkan margin $\frac{2}{\|\mathbf{w}\|}$, yang secara matematis diformulasikan sebagai permasalahan optimasi:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{terhadap kendala} \quad y_i(\mathbf{w}^T x_i + b) \geq 1, \ \forall i \tag{2.13}$$

Pada kasus data yang tidak sepenuhnya dapat dipisahkan secara linear, SVM memperkenalkan variabel *slack* ξ_i dan parameter regularisasi C untuk mentoleransi sejumlah kesalahan klasifikasi, sehingga fungsi objektif menjadi $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i$, di mana nilai C yang lebih besar memberikan penalti lebih tinggi terhadap kesalahan klasifikasi namun berisiko *overfitting* [47]. Dalam penelitian ini, SVM digunakan sebagai *meta-learner* pada pendekatan *stacking ensemble*, di mana SVM mempelajari *hyperplane* optimal berdasarkan probabilitas prediksi dari model-model dasar (*base models*).

2.9.1 Kernel Trick pada SVM

Untuk data yang tidak dapat dipisahkan secara linear pada ruang fitur aslinya, SVM memanfaatkan *kernel trick*, yaitu teknik yang memungkinkan SVM menemukan *hyperplane* pemisah pada ruang fitur berdimensi lebih tinggi tanpa perlu menghitung transformasi fitur secara eksplisit [47, 48]. Ide dasar *kernel trick* adalah bahwa proses pelatihan dan prediksi SVM hanya bergantung pada hasil kali dalam (*inner product*) antar pasangan data, $x_i \cdot x_j$, sehingga hasil kali dalam tersebut dapat digantikan dengan sebuah fungsi *kernel* $K(x_i, x_j)$ yang secara implisit merepresentasikan hasil kali dalam pada ruang fitur berdimensi tinggi $\phi(x)$:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j) \tag{2.14}$$

Dengan pendekatan ini, SVM dapat mempelajari batas keputusan non-linear pada ruang fitur asli tanpa perlu menghitung $\phi(x)$ secara eksplisit, yang dapat berdimensi sangat tinggi atau bahkan tak hingga, sehingga secara signifikan mengurangi kompleksitas komputasi [48].

Beberapa fungsi *kernel* yang umum digunakan antara lain *kernel* linear ($K(x_i, x_j) = x_i \cdot x_j$), *kernel* polinomial, dan *Radial Basis Function* (RBF) *kernel*, yang dirumuskan sebagai:

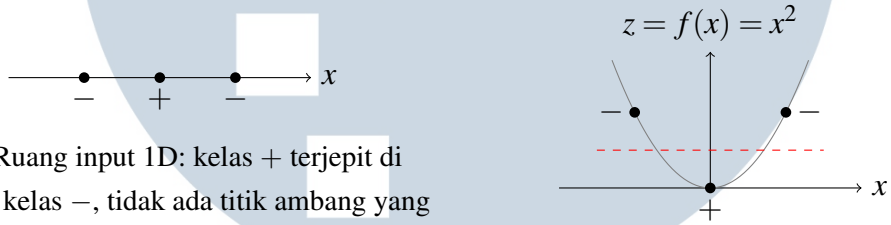
$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{2.15}$$

di mana γ adalah parameter yang mengontrol seberapa jauh pengaruh satu titik data terhadap batas keputusan; nilai γ yang besar menghasilkan batas keputusan yang lebih kompleks dan sensitif terhadap data lokal, sedangkan nilai γ yang kecil menghasilkan batas keputusan yang lebih halus (*smooth*).

Sebagai ilustrasi sederhana dari konsep *kernel trick*, perhatikan permasalahan klasifikasi biner satu dimensi berikut. Misalkan terdapat tiga titik data pada garis x: satu titik kelas positif di $x = 0$, dan dua titik kelas negatif di $x = -1$ dan $x = 1$, sebagaimana ditunjukkan pada Gambar 2.3(a). Karena titik kelas positif berada tepat di antara kedua titik kelas negatif, tidak terdapat satu titik ambang pun pada garis 1D yang dapat memisahkan kelas positif dari kelas negatif, sehingga data ini tidak dapat dipisahkan secara linear pada ruang input aslinya.

Dengan menerapkan fungsi transformasi sederhana $f(x) = x^2$, diperoleh

$f(-1) = 1$, $f(0) = 0$, dan $f(1) = 1$. Pada ruang fitur baru yang dihasilkan oleh f, titik kelas positif berada pada $z = 0$ sedangkan kedua titik kelas negatif berada pada $z = 1$, sehingga kedua kelas kini dapat dipisahkan secara linear hanya dengan satu nilai ambang, misalnya $z = 0{,}5$, sebagaimana ditunjukkan pada Gambar 2.3(b). Fungsi $f(x) = x^2$ pada ilustrasi ini berperan sebagai pemetaan fitur $\phi(x)$ yang mendasari konsep *kernel*, di mana kernel trick secara spesifik merujuk pada kemampuan menghitung hasil kali dalam pada ruang fitur baru tersebut, $K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$, tanpa perlu menghitung $\phi(x)$ secara eksplisit, sebagaimana dirumuskan pada Persamaan (2.14). Ilustrasi ini menggambarkan prinsip dasar di balik kernel trick: data yang tidak dapat dipisahkan secara linear pada ruang input dapat menjadi dapat dipisahkan secara linear setelah dipetakan ke ruang fitur baru.



(a) Ruang input 1D: kelas + terjepit di antara kelas −, tidak ada titik ambang yang memisahkan keduanya.

(b) Setelah transformasi $f(x) = x^2$, kedua kelas terpisah secara linear pada $z = 0{,}5$.

Gambar 2.3. Ilustrasi sederhana *kernel trick*: data yang tidak dapat dipisahkan secara linear pada ruang input satu dimensi (a) menjadi dapat dipisahkan secara linear setelah dipetakan menggunakan $f(x) = x^2$ ke ruang fitur baru (b).

Pemilihan fungsi *kernel* dan parameternya secara langsung memengaruhi kompleksitas model dan kemampuan generalisasinya, sehingga umumnya ditentukan melalui pencarian *hyperparameter* seperti *grid search* pada data validasi.

2.10 Random Forest

Random Forest merupakan algoritma *ensemble learning* berbasis *decision tree* yang diperkenalkan oleh Breiman [49]. Algoritma ini membangun sejumlah besar pohon keputusan (*decision trees*) secara independen menggunakan teknik *bootstrap aggregating* (*bagging*), di mana setiap pohon dilatih pada subset data yang diambil secara acak dengan pengembalian (*sampling with replacement*) dari data pelatihan asli. Selain itu, pada setiap pemisahan (*split*) node dalam pohon, hanya subset acak dari fitur yang dipertimbangkan, sehingga meningkatkan keberagaman (*diversity*) antar pohon dan mengurangi korelasi antar prediksi.

Prediksi akhir Random Forest untuk tugas klasifikasi diperoleh melalui *majority voting* dari seluruh pohon dalam *forest*:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), \ldots, h_T(x)\} \tag{2.16}$$

di mana $h_t(x)$ adalah prediksi dari pohon ke-t dan T adalah jumlah total pohon dalam *forest*. Kombinasi *bagging* dan seleksi fitur acak ini menjadikan Random Forest relatif tahan terhadap *overfitting* dibandingkan *decision tree* tunggal, serta mampu menangani data berdimensi tinggi seperti representasi TF-IDF pada tugas klasifikasi teks [49].

2.11 Logistic Regression

Logistic Regression merupakan algoritma klasifikasi linear yang memodelkan probabilitas suatu instance termasuk ke dalam kelas tertentu menggunakan fungsi *sigmoid* [50]. Berbeda dengan regresi linear yang menghasilkan output kontinu tanpa batas, Logistic Regression mentransformasikan kombinasi linear dari fitur input ke dalam rentang probabilitas $[0, 1]$ melalui fungsi *sigmoid*:

$$P(y = 1 \mid x) = \sigma(\mathbf{w}^T x + b) = \frac{1}{1 + e^{-(\mathbf{w}^T x + b)}} \tag{2.17}$$

di mana $\mathbf{w}$ adalah vektor bobot fitur, b adalah bias, dan $\sigma(\cdot)$ adalah fungsi *sigmoid*. Parameter $\mathbf{w}$ dan b dipelajari melalui minimisasi fungsi *cross-entropy loss* menggunakan metode optimasi berbasis gradien, sebagaimana ditunjukkan pada Persamaan (2.9). Meskipun sederhana dan mudah diinterpretasikan karena setiap bobot fitur merepresentasikan kontribusinya terhadap probabilitas kelas, Logistic Regression memiliki keterbatasan dalam menangkap hubungan non-linear antar fitur, berbeda dengan algoritma seperti Random Forest maupun model berbasis *deep learning* [50].

2.12 K-Fold Cross Validation

K-fold cross validation merupakan teknik evaluasi model yang membagi dataset menjadi k subset (fold) yang sama besar. Pada setiap iterasi, satu fold

digunakan sebagai data uji sementara $k-1$ fold sisanya digunakan sebagai data latih, sehingga setiap sampel digunakan tepat satu kali sebagai data uji [51, 52]. Proses ini diulang sebanyak k kali dan metrik evaluasi akhir dihitung sebagai rata-rata dari seluruh iterasi:

$$\text{CV Score} = \frac{1}{k}\sum_{i=1}^{k}\text{Score}_i \tag{2.18}$$

Keunggulan k-fold cross validation adalah kemampuannya dalam memberikan estimasi performa yang lebih stabil dan tidak bergantung pada satu pembagian data tertentu, sehingga mengurangi varians estimasi dibandingkan single holdout split [53]. Nilai $k = 5$ umum digunakan karena menawarkan keseimbangan antara bias estimasi dan biaya komputasi.

Dalam konteks *stacking ensemble*, *k-fold cross validation* digunakan untuk menghasilkan *out-of-fold (OOF) predictions*, yaitu prediksi pada data yang tidak digunakan selama pelatihan model pada fold tersebut [54]. OOF predictions kemudian digunakan sebagai fitur untuk melatih *meta-learner*, sehingga *meta-learner* tidak terpapar pada data yang sama dengan yang digunakan untuk melatih base models dan risiko data leakage dapat diminimalkan.

2.13 Hyperparameter Optimization dengan Optuna

Hyperparameter optimization merupakan proses pencarian konfigurasi *hyperparameter* yang menghasilkan performa model terbaik pada suatu tugas tertentu. Berbeda dengan parameter model yang dipelajari selama pelatihan, *hyperparameter* seperti *learning rate*, *batch size*, dan *warmup ratio* ditetapkan sebelum proses pelatihan dimulai dan memiliki pengaruh signifikan terhadap kualitas model yang dihasilkan.

Optuna merupakan framework *hyperparameter* optimization otomatis yang dikembangkan oleh Akiba et al. [55] dengan pendekatan define-by-run, di mana ruang pencarian *hyperparameter* didefinisikan secara dinamis selama proses pencarian berlangsung. Berbeda dengan metode grid search yang menguji seluruh kombinasi *hyperparameter* secara exhaustive, Optuna menggunakan algoritma Tree-structured Parzen Estimator (TPE) untuk memprioritaskan kombinasi *hyperparameter* yang lebih menjanjikan berdasarkan hasil trial sebelumnya, sehingga pencarian dapat dilakukan secara lebih efisien pada ruang pencarian yang

luas.

Pada setiap trial, TPE membangun model probabilistik yang memisahkan konfigurasi *hyperparameter* yang menghasilkan performa baik dan buruk berdasarkan riwayat trial sebelumnya. Konfigurasi baru kemudian dipilih dengan memaksimalkan rasio probabilitas antara kedua kelompok tersebut, sehingga pencarian secara bertahap diarahkan menuju wilayah ruang *hyperparameter* yang lebih menjanjikan [55].

Untuk meningkatkan efisiensi pencarian lebih lanjut, Optuna menyediakan mekanisme *pruning* yang menghentikan trial lebih awal apabila performa pada tahap awal pelatihan menunjukkan hasil yang tidak menjanjikan. Salah satu strategi pruning yang tersedia adalah MedianPruner, yang menghentikan suatu trial apabila performa intermedietnya berada di bawah median performa trial-trial sebelumnya pada tahap yang sama. Kombinasi antara TPE dan pruning memungkinkan Optuna untuk menemukan konfigurasi *hyperparameter* yang optimal dengan jumlah trial yang lebih sedikit dibandingkan pendekatan pencarian konvensional [55].

2.14 Explainable AI dan LIME

Explainable Artificial Intelligence (XAI) merupakan bidang yang berfokus pada pengembangan metode dan teknik untuk membuat keputusan model machine learning dapat dipahami dan diinterpretasikan oleh manusia [56, 57]. Kebutuhan akan XAI semakin meningkat seiring dengan meluasnya penggunaan model deep learning yang bersifat black-box, di mana proses pengambilan keputusan internal model tidak dapat langsung diamati [58].

Local Interpretable Model-Agnostic Explanations (LIME) merupakan salah satu metode XAI yang diperkenalkan oleh Ribeiro et al. [58] untuk menjelaskan prediksi model klasifikasi secara lokal dan model-agnostic. LIME bekerja dengan cara menghasilkan sampel data di sekitar instance yang ingin dijelaskan, memperoleh prediksi model untuk setiap sampel tersebut, kemudian melatih model interpretabel yang lebih sederhana (seperti regresi linear) pada sampel-sampel tersebut untuk mendekati perilaku model black-box secara lokal [58].

Secara formal, LIME mencari penjelasan g dari ruang model interpretabel G yang meminimalkan kompleksitas $\Omega(g)$ sambil memaksimalkan ketepatan pendekatan lokal [58]:

$$\xi(x) = \arg\min_{g \in G} \mathscr{L}(f, g, \pi_x) + \Omega(g) \tag{2.19}$$

di mana f adalah model yang akan dijelaskan, π_x adalah fungsi kedekatan (proximity) yang mendefinisikan neighborhood di sekitar instance x, dan $\mathscr{L}$ mengukur seberapa baik g mendekati f dalam neighborhood tersebut [58].

Dalam konteks klasifikasi teks, LIME mengidentifikasi token atau kata yang paling berkontribusi terhadap prediksi model untuk suatu instance tertentu [58]. Pendekatan ini memungkinkan peneliti untuk memverifikasi apakah model menggunakan fitur-fitur yang relevan secara semantik dalam mengambil keputusan, sehingga meningkatkan kepercayaan terhadap model dan membantu mengidentifikasi potensi bias [17, 58].

2.15 Metrik Evaluasi

Evaluasi performa model klasifikasi dalam penelitian ini menggunakan beberapa metrik yang diturunkan dari confusion matrix [59]. Confusion matrix merangkum hasil prediksi model dalam empat komponen: *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)*, dan *False Negative (FN)*.

2.15.1 Akurasi

Akurasi mengukur proporsi prediksi yang benar dari seluruh prediksi yang dibuat oleh model [59]:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.20}$$

Meskipun akurasi merupakan metrik yang intuitif, metrik ini dapat menyesatkan pada dataset yang tidak seimbang karena model yang hanya memprediksi kelas mayoritas dapat memperoleh akurasi tinggi.

2.15.2 Presisi

Presisi mengukur proporsi prediksi positif yang benar dari seluruh prediksi positif yang dibuat model [59]:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.21}$$

Presisi yang tinggi menunjukkan bahwa model jarang mengklasifikasikan teks yang bukan ujaran kebencian sebagai ujaran kebencian (*false positive* rendah).

2.15.3 Recall

Recall mengukur proporsi instance positif yang berhasil diidentifikasi oleh model dari seluruh instance positif yang sebenarnya [59]:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.22}$$

Recall yang tinggi menunjukkan bahwa model mampu mendeteksi sebagian besar ujaran kebencian yang ada (*false negative* rendah), yang merupakan aspek penting dalam sistem deteksi konten berbahaya.

2.15.4 F1-Score

F1-score merupakan rata-rata harmonik dari presisi dan recall, sehingga memberikan keseimbangan antara kedua metrik tersebut [59]:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.23}$$

F1-score digunakan sebagai metrik utama dalam penelitian ini karena lebih informatif dibandingkan akurasi pada dataset dengan distribusi kelas yang seimbang namun tetap perlu mempertimbangkan trade-off antara presisi dan recall.

2.15.5 Confusion Matrix

Confusion matrix menyajikan ringkasan visual dan numerik dari distribusi prediksi model pada setiap kombinasi kelas aktual dan kelas prediksi [60]. Dalam konteks klasifikasi biner ujaran kebencian, confusion matrix terdiri atas empat sel: TP (ujaran kebencian diprediksi benar), TN (non-ujaran kebencian diprediksi benar), FP (non-ujaran kebencian salah diprediksi sebagai ujaran kebencian), dan FN (ujaran kebencian salah diprediksi sebagai non-ujaran kebencian). Analisis confusion matrix memungkinkan identifikasi pola kesalahan prediksi model secara lebih mendalam dibandingkan metrik agregat.