Metodologi

Studi ini bertujuan untuk mengklasifikasikan komentar-komentar toksik ke dalam berbagai kategori, termasuk pornografi, SARA, radikalisme, dan pencemaran nama baik, menggunakan berbagai model pembelajaran mesin dan pembelajaran mendalam. Metodologi yang diterapkan oleh peneliti melibatkan lima langkah utama: akuisisi dataset, pra-pemrosesan data, penerapan ekstraksi fitur, formulasi dan pelatihan model, serta evaluasi kinerjanya.

1) Dataset

Studi ini menggunakan kumpulan data besar berisi komentar-komentar yang kasar dari berbagai media sosial di Indonesia. Data ini diambil dari proyek "Netifier: Negativity Classifier," milik Ahmad Izzan, Christian Wibisono, dan Ilham Firdausi Putra. yang bertujuan membangun sistem canggih untuk mengelompokkan teks kasar berbahasa Indonesia. Pentingnya data ini adalah karena sangat sulit menemukan data publik yang cocok untuk menganalisis teks kasar dalam Bahasa Indonesia.

Sumber dan Proses Pengumpulan Data

Data dikumpulkan dengan cara mengambil langsung dari beberapa platform media sosial terkenal di Indonesia, seperti Instagram, Twitter, dan Kaskus. Setelah data terkumpul, sekitar 7.000 sampel pertama diberi label secara manual oleh tim proyek. Mereka membagi komentar-komentar ini ke dalam empat jenis kekasaran yang berbeda.

Struktur Dataset

Setiap entri dalam dataset dirancang untuk menyertakan informasi rinci mengenai komentar, yang terdiri dari kolom-kolom berikut:

a) original_text: Berisi teks komentar asli yang belum mengalami modifikasi atau pembersihan.
b) source: Menunjukkan platform media sosial asal komentar (misalnya, 'kaskus', 'instagram', 'twitter').
c) pornografi: Label biner (1 untuk komentar yang mengandung pornografi, 0 untuk yang tidak).
d) sara: Label biner (1 untuk komentar yang mengandung konten SARA, 0 untuk yang tidak).
e) radikalisme: Label biner (1 untuk komentar yang mengandung radikalisme, 0 untuk yang tidak).
f) pencemaran_nama_baik: Label biner (1 untuk komentar yang mengandung pencemaran nama baik, 0 untuk yang tidak).

Desain *multi-label* ini memungkinkan satu komentar untuk diklasifikasikan ke dalam satu atau lebih kategori toksik secara simultan, mencerminkan sifat kompleks toksisitas dalam skenario dunia nyata.

Statistik dan Wawasan Eksplorasi Dataset

Dataset dibagi menjadi dua partisi utama: satu untuk pelatihan model dan satu lagi untuk evaluasi kinerja model. Partisi pelatihan terdiri dari 6.995 *instance* komentar, sedangkan partisi pengujian mencakup 778 *instance*.

Analisis distribusi label toksisitas menunjukkan konsistensi signifikan di kedua partisi data, mengindikasikan pembagian data yang seimbang dan representatif:

Distribusi Label pada Data Pelatihan:

a) Pornografi: 22,50%
b) SARA: 16,07%
c) Radikalisme: 16,50%
d) Pencemaran Nama Baik: 31,17%

Distribusi Label pada Data Pengujian:

a) Pornografi: 22,24%
b) SARA: 14,65%
c) Radikalisme: 15,68%
d) Pencemaran Nama Baik: 31,49%

Komentar Toksik di Setiap Platform

Tabel dan grafik di bawah ini menggambarkan distribusi komentar toksik berdasarkan platform (Instagram, Kaskus, Twitter) dan jenis toksisitas.

Source	Pornografi	SARA	Radikalisme	Pencemaran Nama Baik
Instagram	277	6	0	585
Kaskus	154	385	139	602
Twitter	1143	733	1015	993

Toxic Label Combination Frequency

The following table and chart show the frequency of different combinations of toxicity labels present in the dataset, ordered from most frequent to least frequent.

Kombinasi Label	Frekuensi
sara, radikalisme, pencemaran_nama_baik	548
pornografi, pencemaran_nama_baik	429
sara, pencemaran_nama_baik	309
radikalisme, pencemaran_nama_baik	95
sara, radikalisme	72
pornografi, sara, pencemaran_nama_baik	35
pornografi, radikalisme, pencemaran_nama_baik	9
pornografi, sara, radikalism	5
pornografi, sara	3
pornografi, sara	1

Kata-kata Toksik Teratas per Kategori

Berikut adalah 10 kata toksik teratas untuk setiap label, beserta frekuensinya

1) Pornografi

Kata	Frekuensi
k*ntol	795
m*mek	737
t*tek	363
agama	340
remas	321
susu	218

ingin	214
s*nge	202
cewek	201
s*ngama	199

2) SARA

Kata	Frekuensi
kafir	536
cina	448
agama	355
islam	348
bom	312
indonesia	278
negara	214
muslim	193
ganyang	188
papua	161

3) Radikalisme

Kata	Frekuensi
kafir	653
islam	380
bom	355
khilafah	346
cina	313
indonesia	266

bunuh	257
agama	238
negara	224
usir	214

4) Pencemaran Nama Baik

Kata	Frekuensi
agama	565
kafir	551
cina	373
k*ntol	344
bom	339
bodoh	325
islam	296
muka	226
banci	225
indonesia	223

Analisis ini memberikan pemahaman mendalam tentang karakteristik dataset, termasuk komposisi, distribusi label, sebaran di berbagai platform, dan identifikasi kata-kata kunci toksik, yang sangat penting untuk langkah selanjutnya.

2) Pra-pemrosesan Data dan Ekstraksi Fitur

Setelah memuat data mentah, pipeline pra-pemrosesan yang komprehensif diterapkan untuk membersihkan dan menstandarisasi data teks. Langkah krusial ini memastikan bahwa data teks berada dalam format yang sesuai untuk ekstraksi fitur dan pelatihan model yang efektif.

Wawasan dari Analisis Data Eksplorasi (ADE):

ADE awal dilakukan untuk memahami karakteristik dan kualitas data teks mentah. Observasi kunci yang menginformasikan keputusan pra-pemrosesan meliputi:

a) **Keberadaan Pemformatan Spesifik Platform**: Banyak komentar berasal dari platform media sosial seperti Twitter dan Instagram, serta forum seperti Kaskus. Hal ini terbukti dengan adanya pola seperti @username (sebutan), URL yang tidak relevan, dan kode pemformatan Kaskus seperti [QUOTE...] dan entitas HTML ". Kehadiran format ini merupakan "noise" yang dapat mengganggu analisis tekstual dan perlu dihilangkan.
b) **Penggunaan Bahasa Informal dan Gaul**: Komentar seringkali mengandung kata-kata gaul dan singkatan (slang words) yang umum dalam percakapan sehari-hari. Contohnya, "yg" untuk "yang", "gitu" untuk "begitu". Bahasa informal ini perlu distandarisasi agar model dapat memahami makna yang sebenarnya.
c) **Anomali Karakter dan Spasi**: Ditemukan adanya karakter baris baru (\n) yang berlebihan, spasi ganda atau lebih (excessive whitespace), serta pengulangan karakter yang berlebihan (misalnya, "haaaaiii" yang seharusnya "haai"). Anomali ini dapat menyebabkan tokenisasi yang tidak akurat dan menambah kompleksitas yang tidak perlu bagi model.
d) **Keberadaan Emotikon dan Simbol**: Emotikon (misalnya, :), :() digunakan untuk mengekspresikan sentimen. Meskipun mereka mengandung informasi, formatnya tidak langsung dapat diproses oleh model teks. Selain itu, ada juga karakter non-alfabetik lainnya dan simbol yang tidak berkontribusi pada makna tekstual.

Wawasan dari ADE ini menegaskan perlunya serangkaian langkah pra-pemrosesan yang teliti untuk mengubah teks mentah menjadi representasi yang bersih, konsisten, dan siap untuk dimodelkan.

Pipeline Pra-pemrosesan Teks:

Langkah-langkah pra-pemrosesan, yang diterapkan secara berurutan dalam fungsi preprocess_text, adalah sebagai berikut:

a) **Konversi Huruf Kecil (text.lower()):** Semua karakter dalam original_text dikonversi menjadi huruf kecil. Langkah ini menormalkan kapitalisasi, memastikan bahwa kata-kata seperti "Toxic" dan "toxic" diperlakukan sebagai token yang sama.
b) **Penghapusan Karakter Baris Baru (remove_newline):** Karakter baris baru (\n) diganti dengan spasi tunggal. Ini meratakan multi-baris teks menjadi satu baris, menghilangkan pemformatan struktural yang tidak relevan untuk analisis konten.
c) **Penghapusan URL (remove_url):** Ekspresi reguler digunakan untuk mendeteksi dan menghapus URL (misalnya, www., https://, http://). URL biasanya tidak membawa makna semantik untuk klasifikasi toksisitas dan dapat menimbulkan kebisingan.
d) **Penghapusan Pemformatan Twitter/Instagram (remove_twitter_ig_formatting):** Pola seperti @username (sebutan Twitter/Instagram) dihapus. Ini adalah pengidentifikasi spesifik platform daripada kata-kata yang mengandung konten.

e) **Penghapusan Pemformatan Kaskus (remove_kaskus_formatting):** Fungsi ini menargetkan dan menghapus artefak forum Kaskus tertentu seperti blok [quote...]...[/quote], tag kurung lainnya [...], dan entitas HTML ". Ini membersihkan markup khusus forum.
f) **Terjemahan Emotikon (translate_emoticon):** Emotikon (misalnya, :), :() diganti dengan kata-kata sentimen yang sesuai (misalnya, "senang", "sedih") menggunakan kamus yang dimuat dari emoticon.txt. Ini memperkaya teks dengan ekspresi sentimen eksplisit.
g) **Tokenisasi Teks (tokenize_text):** WordPunctTokenizer dari NLTK digunakan untuk membagi teks menjadi kata-kata dan tanda baca. Parameter opsional memungkinkan penghapusan tanda baca, hanya mempertahankan token alfanumerik. Token kemudian digabungkan kembali menjadi string yang dipisahkan spasi.
h) **Transformasi Kata-kata Gaul (transform_slang_words):** Kata-kata gaul informal dan singkatan dikonversi menjadi padanan baku dalam bahasa Indonesia menggunakan kamus yang berasal dari slangword.csv. Ini menstandarisasi penggunaan bahasa informal (misalnya, "yg" menjadi "yang"). Fungsi ini diterapkan dua kali, sekali sebelum dan sekali setelah penghapusan karakter, untuk memastikan transformasi slang yang komprehensif, karena penghapusan karakter mungkin secara tidak sengaja menciptakan pola seperti slang baru.
i) **Penghapusan Karakter Berulang (remove_repeating_characters):** Fungsi ini mengurangi urutan tiga atau lebih karakter identik menjadi paling banyak dua pengulangan (misalnya, "hahahaha" menjadi "hahaha"). Ini menormalkan perpanjangan kata yang umum dalam penulisan informal sambil tetap mempertahankan beberapa penekanan.
j) **Penghapusan Karakter Non-Alfabet (remove_non_alphabet):** Semua karakter yang bukan alfabet Inggris (a-z, A-Z) atau spasi dihapus. Ini memastikan bahwa hanya konten tekstual yang relevan yang tersisa.
k) **Penghapusan Spasi Berlebihan (remove_excessive_whitespace):** Beberapa spasi berturut-turut digabungkan menjadi satu spasi tunggal. Ini menstandarisasi spasi dan mencegah kesalahan tokenisasi dari spasi ekstra.
l) **Penghapusan Spasi Awal/Akhir (Strip Whitespace):** Terakhir, spasi di awal dan akhir teks yang telah diproses dihapus.

Setelah pra-pemrosesan, teks yang bersih disimpan dalam kolom baru bernama processed_text di kedua dataset pelatihan dan pengujian, yang kemudian disimpan sebagai processed_train.csv dan processed_test.csv untuk penggunaan selanjutnya.

Ekstraksi Fitur TF-IDF

Untuk ekstraksi fitur, TfidfVectorizer diterapkan untuk mengubah teks yang telah diproses menjadi representasi numerik. TF-IDF (Term Frequency-Inverse Document Frequency)

menetapkan bobot pada setiap kata berdasarkan frekuensinya dalam sebuah dokumen dan kelangkaannya di seluruh korpus. Pendekatan ini membantu menyoroti kata-kata yang penting untuk dokumen tertentu tetapi tidak terlalu umum di semua dokumen, menjadikannya efektif untuk klasifikasi teks.

TfidfVectorizer dikonfigurasi dengan max_features=10000, membatasi kosakata menjadi 10.000 kata yang paling sering muncul. Selain itu, daftar lengkap stop words bahasa Indonesia (misalnya, 'sih', 'ya', 'dan', 'yang', 'tidak') disediakan untuk vectorizer. Kata-kata umum ini, yang biasanya memiliki sedikit makna semantik, dihapus, sehingga mengurangi dimensi dan meningkatkan fokus model pada istilah yang lebih diskriminatif. Vectorizer ini di-fit secara eksklusif pada data pelatihan (X_train) untuk mempelajari kosakata dan frekuensi dokumen terbalik, dan kemudian digunakan untuk mengubah kedua data pelatihan (X_train_tfidf) dan pengujian (X_test_tfidf) menjadi matriks sparse. Matriks X_train_tfidf yang dihasilkan memiliki bentuk (6995, 10000), mengkonfirmasi keberhasilan vectorisasi data pelatihan menjadi 10.000 fitur.

3) Pelatihan & Evaluasi Model: Regresi Logistik

Regresi Logistik adalah model linier fundamental yang banyak digunakan untuk tugas klasifikasi. Meskipun namanya demikian, model ini memprediksi probabilitas suatu instance termasuk dalam kelas tertentu dengan menerapkan fungsi logistik (sigmoid) ke kombinasi linier fitur input. Untuk klasifikasi multi-label, di mana satu instance dapat memiliki beberapa label benar, Regresi Logistik dapat diperluas dengan melatih pengklasifikasi biner independen untuk setiap label.

Untuk tugas klasifikasi multi-label ini, LogisticRegression dibungkus dengan MultiOutputClassifier. Ini berarti pengklasifikasi Regresi Logistik terpisah dilatih untuk setiap empat label target: pornografi, sara, radikalisme, dan pencemaran_nama_baik.

a) **Parameter**: Model LogisticRegression diinisialisasi dengan max_iter=1000, menetapkan jumlah iterasi maksimum untuk algoritma optimasi agar konvergen. Yang penting, class_weight='balanced' digunakan. Parameter ini secara otomatis menyesuaikan bobot secara berbanding terbalik dengan frekuensi kelas, yang sangat penting untuk menangani ketidakseimbangan kelas yang melekat yang sering diamati dalam dataset multi-label, memastikan bahwa kelas minoritas tidak diabaikan selama pelatihan.
b) **Pelatihan**: Model multi_output_logreg dilatih dengan mem-fit-nya ke data pelatihan TF-IDF yang telah di-vektorisasi (X_train_tfidf) dan target multi-label yang sesuai (y_train).
c) **Evaluasi**: Setelah fase pelatihan, prediksi (logreg_preds) dihasilkan pada data X_test_tfidf yang tidak terlihat menggunakan metode predict. Kinerja model kemudian dinilai secara kuantitatif menggunakan dua metrik utama: accuracy_score untuk memberikan akurasi kecocokan persis secara keseluruhan, dan classification_report.

Laporan klasifikasi menawarkan rincian detail presisi, recall, dan F1-score untuk setiap label individu, memberikan wawasan tentang kinerja model di berbagai kategori toksisitas.

4) Pelatihan & Evaluasi Model: SVC Linier

Klasifikasi Support Vector Linier (Linear SVC) adalah varian yang kuat dan efisien dari Support Vector Machines (SVM) yang dirancang khusus untuk dataset besar dengan banyak fitur. Model ini bertujuan untuk menemukan batas keputusan linier (hyperplane) yang paling baik memisahkan kelas-kelas yang berbeda dalam ruang fitur, memaksimalkan margin di antara mereka. Linear SVC sangat cocok untuk data berdimensi tinggi, seperti fitur TF-IDF yang dihasilkan dari teks.

Serupa dengan Regresi Logistik, LinearSVC diintegrasikan dengan MultiOutputClassifier untuk secara efektif menangani sifat multi-label dari tugas klasifikasi. Pengaturan ini melibatkan pelatihan model Linear SVC independen untuk setiap label keluaran.

a) **Parameter**: Model LinearSVC dikonfigurasi dengan max_iter=1000 untuk memastikan bahwa algoritma optimasi mencapai konvergensi dalam jumlah langkah yang wajar. Parameter class_weight='balanced' diterapkan untuk mengatasi potensi ketidakseimbangan kelas, memberikan bobot yang lebih tinggi pada kelas minoritas selama pelatihan. Selain itu, dual=False ditentukan. Pengaturan ini seringkali lebih efisien untuk dataset di mana jumlah sampel (n_samples) jauh lebih besar daripada jumlah fitur (n_features), seperti yang umumnya terjadi setelah vektorisasi TF-IDF.
b) **Pelatihan**: Model multi_output_linearsvc dilatih dengan mem-fit-nya ke data pelatihan TF-IDF yang telah di-vektorisasi (X_train_tfidf) dan variabel target multi-label yang sesuai (y_train).
c) **Evaluasi**: Setelah pelatihan selesai, prediksi (linearsvc_preds) dihasilkan pada dataset X_test_tfidf. Kinerja model kemudian dievaluasi secara kuantitatif menggunakan accuracy_score untuk akurasi kecocokan persis secara keseluruhan dan classification_report yang komprehensif untuk merinci presisi, recall, dan F1-score untuk masing-masing dari empat label toksisitas.

5) Pelatihan & Evaluasi Model: Random Forest

Random Forest adalah metode pembelajaran ensemble yang membangun banyak pohon keputusan selama pelatihan dan menghasilkan mode kelas (untuk klasifikasi) dari pohon-pohon individu. Model ini beroperasi dengan membangun pohon keputusan yang beragam, masing-masing dilatih pada subset data yang di-bootstrap dan subset fitur acak. "Keacakan" ini membantu mendekorelasikan pohon-pohon dan mengurangi overfitting, yang mengarah pada peningkatan kinerja generalisasi dan kekokohan.

Untuk masalah klasifikasi multi-label ini, RandomForestClassifier digunakan bersama dengan MultiOutputClassifier. Pendekatan ini melibatkan pelatihan RandomForestClassifier independen untuk setiap dari empat label toksisitas.

a) **Parameter**: RandomForestClassifier diinisialisasi dengan n_estimators=100, menunjukkan bahwa 100 individu pohon keputusan akan dibangun di dalam forest. random_state=42 diatur untuk memastikan reproduktibilitas hasil di berbagai eksekusi. Serupa dengan model lain, class_weight='balanced' digunakan untuk menangani ketidakseimbangan kelas dalam setiap label, memastikan bahwa model memberikan pertimbangan yang tepat untuk semua kelas.
b) **Pelatihan**: Model multi_output_rf dilatih dengan mem-fit-nya ke fitur pelatihan TF-IDF yang telah di-vektorisasi (X_train_tfidf) dan array target multi-label (y_train).
c) **Evaluasi**: Setelah pelatihan, metode predict digunakan untuk menghasilkan prediksi (rf_preds) pada dataset X_test_tfidf. Kinerja model kemudian dievaluasi dengan menghitung accuracy_score untuk akurasi kecocokan persis secara keseluruhan dan menghasilkan classification_report untuk memberikan metrik detail (presisi, recall, F1-score) untuk setiap kategori toksisitas individu.

6) Pelatihan & Evaluasi Model: XGBoost

XGBoost (Extreme Gradient Boosting) adalah pustaka gradient boosting yang sangat dioptimalkan, terdistribusi, dan fleksibel yang dirancang untuk efisiensi dan kinerja. Ini adalah metode pembelajaran ensemble yang secara berurutan membangun pohon keputusan, di mana setiap pohon baru memperbaiki kesalahan dari pohon-pohon sebelumnya. XGBoost secara luas dikenal karena kecepatan, skalabilitas, dan hasil canggihnya di berbagai tantangan pembelajaran mesin, termasuk klasifikasi.

Untuk tugas klasifikasi multi-label, xgb.XGBClassifier dibungkus dalam MultiOutputClassifier. Ini berarti model XGBoost terpisah dilatih untuk setiap dari empat label toksisitas yang berbeda.

a) **Parameter**: xgb.XGBClassifier dikonfigurasi dengan random_state=42 untuk memastikan reproduktibilitas proses pelatihan. eval_metric='logloss' ditentukan, yang merupakan metrik evaluasi umum dan kuat untuk masalah klasifikasi, terutama ketika berhadapan dengan keluaran probabilistik, karena mengukur akurasi pengklasifikasi dengan menghukum klasifikasi yang salah berdasarkan kepercayaan prediksi.
b) **Pelatihan**: Model multi_output_xgboost dilatih dengan memanggil metode fit-nya dengan data pelatihan TF-IDF yang telah di-vektorisasi (X_train_tfidf) dan target multi-label (y_train).
c) **Evaluasi**: Setelah pelatihan, prediksi (xgboost_preds) dihasilkan pada dataset X_test_tfidf yang tidak terlihat menggunakan metode predict. Kinerja model secara keseluruhan dikuantifikasi menggunakan accuracy_score untuk akurasi kecocokan persis,

dan classification_report yang detail dihasilkan untuk memberikan metrik per-label, termasuk presisi, recall, dan F1-score.

7) Pelatihan & Evaluasi Model: BERT (2 Epoch)

BERT (Bidirectional Encoder Representations from Transformers) adalah model bahasa pra-terlatih yang kuat yang telah merevolusi Pemrosesan Bahasa Alami (NLP). Model ini memanfaatkan arsitektur Transformer untuk mempelajari representasi bidireksional mendalam dari sejumlah besar teks yang tidak berlabel dengan mengondisikan secara bersamaan pada konteks kiri dan kanan. Untuk tugas klasifikasi, BERT dapat disetel dengan menambahkan lapisan klasifikasi di atas encoder pra-terlatihnya. Pendekatan ini memungkinkan model untuk memanfaatkan pemahaman linguistiknya yang kaya yang dipelajari selama pra-pelatihan, menjadikannya sangat efektif untuk berbagai tugas NLP hilir, termasuk klasifikasi teks multi-label.

Model bert-base-uncased digunakan untuk tugas klasifikasi teks multi-label ini. Implementasi menggunakan PyTorch dan pustaka Hugging Face Transformers.

a) **Konfigurasi**:
   - MODEL_NAME: 'bert-base-uncased' (varian BERT yang umum digunakan, 'uncased' berarti tidak membedakan antara "Text" dan "text").
   - MAX_LEN: 128 (panjang urutan maksimum untuk tokenisasi). Urutan yang lebih panjang dipotong, yang lebih pendek diisi.
   - BATCH_SIZE: 16 (jumlah sampel yang diproses dalam satu forward/backward pass).
   - EPOCHS: 2 (jumlah pass penuh melalui seluruh dataset pelatihan). Jumlah ini dipilih untuk menyeimbangkan kinerja dan efisiensi komputasi untuk fine-tuning.
   - LEARNING_RATE: 2e-5 (ukuran langkah awal yang digunakan oleh optimizer untuk memperbarui bobot model). Ini adalah learning rate umum untuk fine-tuning model BERT.
b) **Pengaturan Perangkat**: Skrip memeriksa ketersediaan GPU dan mengatur device ke "cuda" jika GPU ada, jika tidak ke "cpu", memastikan optimalnya kecepatan pelatihan.
c) **Tokeniser**: BertTokenizer.from_pretrained(MODEL_NAME) digunakan untuk melakukan tokenisasi teks yang telah diproses sesuai dengan kosakata dan aturan tokenisasi subkata spesifik BERT.
d) **Dataset Kustom (TextDataset)**: Kelas PyTorch Dataset kustom diimplementasikan untuk secara efisien menangani pemuatan dan persiapan teks dan label untuk BERT. Untuk setiap sampel teks, ia melakukan:
   - Mengkodekan teks menggunakan tokeniser BERT, menambahkan token khusus ([CLS], [SEP]).
   - Menerapkan padding ke MAX_LEN dan memotong urutan yang lebih panjang.

    - Menghasilkan mask perhatian untuk membedakan token nyata dari token padding.
    - Mengonversi label ke torch.FloatTensor, yang diperlukan untuk fungsi kerugian BCEWithLogitsLoss.
    - Mengembalikan input_ids, attention_mask, dan labels sebagai tensor PyTorch.
e) **DataLoaders**: Instance DataLoader dibuat untuk train_dataset dan test_dataset. Loader ini mengelola batching dan pengacakan (untuk data pelatihan) data, memfasilitasi iterasi yang efisien selama pelatihan dan evaluasi.
f) **Inisialisasi Model**: BertForSequenceClassification.from_pretrained(MODEL_NAME, num_labels=len(labels), problem_type="multi_label_classification") memuat model BERT pra-terlatih. num_labels diatur ke 4 (untuk empat kategori toksisitas), dan problem_type="multi_label_classification" mengonfigurasi lapisan terakhir model dan perhitungan kerugian untuk keluaran multi-label. Model kemudian dipindahkan ke device (GPU/CPU).
g) **Optimizer dan Scheduler**:
    - AdamW dipilih sebagai optimizer. Ini adalah varian Adam yang menggabungkan weight decay untuk mencegah overfitting, sehingga sangat cocok untuk fine-tuning model besar seperti BERT.
    - get_linear_schedule_with_warmup digunakan sebagai scheduler learning rate. Ini secara linier meningkatkan learning rate dari 0 ke LEARNING_RATE selama fase "warmup" (0 langkah dalam kasus ini, berarti tidak ada warm-up) dan kemudian secara linier menurunkannya ke 0 selama sisa langkah pelatihan.
h) **Fungsi Kerugian**: torch.nn.BCEWithLogitsLoss().to(device) dipilih sebagai fungsi kerugian. Binary Cross-Entropy with Logits Loss ini sesuai untuk klasifikasi multi-label, karena menghitung kerugian secara independen untuk setiap label, langsung pada logits keluaran mentah model sebelum aktivasi sigmoid.
i) **Loop Pelatihan (train_epoch)**:
    - Model diatur ke mode pelatihan (model.train()).
    - Untuk setiap batch, input_ids, attention_mask, dan targets dipindahkan ke device.
    - Forward pass dilakukan melalui model.
    - oss_fn menghitung kerugian antara logits model dan target yang sebenarnya.
    - loss.backward() melakukan backpropagation untuk menghitung gradien.
    - torch.nn.utils.clip_grad_norm_ menerapkan pemotongan gradien (norma maksimum 1.0) untuk mencegah gradien meledak, yang umum terjadi pada jaringan saraf dalam.
    - optimizer.step() memperbarui parameter model berdasarkan gradien yang dihitung.
    - scheduler.step() menyesuaikan learning rate.
    - optimizer.zero_grad() menghapus gradien untuk batch berikutnya.
j) **Fungsi Evaluasi (eval_model)**:

- Model diatur ke mode evaluasi (model.eval()) untuk menonaktifkan dropout dan perilaku khusus pelatihan lainnya.
- torch.no_grad() context manager memastikan bahwa tidak ada gradien yang dihitung, mengurangi konsumsi memori dan mempercepat inferensi.
- Untuk setiap batch, forward pass dilakukan, dan kerugian dihitung.
- Keluaran mentah model (logits) dilewatkan melalui torch.sigmoid untuk mengubahnya menjadi probabilitas (antara 0 dan 1), yang kemudian dikumpulkan bersama dengan target yang sebenarnya.

k) **Prediksi BERT**: Probabilitas yang diperoleh dari fungsi evaluasi (outputs) diubah menjadi prediksi biner (bert_preds) dengan menerapkan ambang batas 0.5. Jika probabilitas yang diprediksi untuk suatu label lebih besar dari 0.5, label tersebut diprediksi sebagai positif (1); jika tidak, itu adalah negatif (0).

l) **Evaluasi**: accuracy_score dan classification_report digunakan untuk mengukur kinerja model BERT pada set pengujian, memberikan akurasi kecocokan persis secara keseluruhan dan metrik per-label.

8) Metrik Evaluasi

Untuk mengevaluasi dan membandingkan kinerja semua model klasifikasi secara menyeluruh, metrik standar berikut digunakan:

a) **Skor Akurasi (Accuracy Score)**: Metrik ini mengukur proporsi instance yang diprediksi dengan benar dari total jumlah instance. Dalam konteks multi-label, ini biasanya mengacu pada *akurasi subset* (atau akurasi kecocokan persis), di mana prediksi untuk sampel tertentu dianggap benar hanya jika *semua* label yang diprediksi sama persis dengan *semua* label yang sebenarnya.

Akurasi=Total Jumlah SampelJumlah Sampel yang Diprediksi dengan Tepat

Jika nilai akurasi meningkat, itu berarti model melakukan lebih banyak prediksi yang benar. Oleh karena itu, semakin tinggi skor akurasi, semakin baik kinerja model dalam memprediksi semua label secara tepat untuk setiap sampel.

b) **Laporan Klasifikasi (Classification Report)**: Laporan detail ini menyediakan metrik kinerja per-kelas, termasuk presisi, recall, F1-score, dan *support*, untuk setiap dari empat label target individu (pornografi, sara, radikalisme, pencemaran_nama_baik).
   - **Presisi (Precision)**: Merepresentasikan proporsi prediksi positif benar di antara semua prediksi positif yang dibuat oleh model untuk kelas tertentu. Ini menjawab pertanyaan: "Dari semua instance yang diprediksi positif untuk kelas ini, berapa banyak yang sebenarnya positif?"

Presisi=True Positives+False PositivesTrue Positives
Semakin tinggi nilai presisi, semakin baik model dalam menghindari false positives (memprediksi positif padahal sebenarnya negatif).

- **Recall (Sensitivitas)**: Merepresentasikan proporsi prediksi positif benar di antara semua instance positif aktual untuk kelas tertentu. Ini menjawab pertanyaan: "Dari semua instance yang sebenarnya positif untuk kelas ini, berapa banyak yang berhasil diidentifikasi dengan benar oleh model?"
  Recall=True Positives+False NegativesTrue Positives
  Semakin tinggi nilai recall, semakin baik model dalam menemukan semua instance positif (menghindari false negatives).
- **F1-Score**: Rata-rata harmonik dari presisi dan recall. Ini memberikan skor tunggal yang menyeimbangkan kedua metrik, membuatnya sangat berguna ketika ada distribusi kelas yang tidak merata atau ketika false positives dan false negatives sama-sama penting untuk dipertimbangkan.
  F1-Score=2×Presisi+RecallPresisi×Recall
  Semakin tinggi nilai F1-score, semakin baik model dalam menyeimbangkan presisi dan recall, memberikan ukuran kinerja yang lebih komprehensif, terutama untuk kelas minoritas.
- **Support**: Mengacu pada jumlah kejadian aktual dari label tertentu dalam dataset pengujian. Misalnya, jika 'pornografi' memiliki support 173, berarti ada 173 komentar di dataset pengujian yang sebenarnya berlabel 'pornografi'. Ini bukan metrik kinerja, melainkan indikator seberapa banyak data yang tersedia untuk kelas tersebut, yang membantu dalam menafsirkan presisi, recall, dan F1-score. Kelas dengan support yang sangat rendah mungkin menunjukkan metrik yang sangat fluktuatif atau tidak stabil.

DIBERIKAN BAB METODOLOGI DIATAS DAN DIBERIKAN 6 PAPER YG MEMILIKI KEMIRIPAN
ANDA SEBAGAI PENELITI DIMINTA MELAKUKAN HAL BERIKUT

1. Baca metodologi
2. Baca semua paper secara keseluruhan
3. Masukan quotes/sitasi ke dalam metodologi dari paper yg diberikan
4. Gunakan quotes format APA yang (Penulis, tahun)
5. Jangan ubah inti konten tapi sambungkan ke paper yg ada tapi jangan menyimpang dr yang dibahas
6. Intinya dalam metodologi harus mencakup 6 paper