BAB 2
LANDASAN TEORI

2.1 Deteksi Sarkasme

Seiring dengan berkembangnya NLP, deteksi sarkasme juga menjadi hal yang esensial dalam menganalisis opini dalam sebuah teks. Kualitas dataset yang digunakan untuk membangun model deteksi sarkasme sangat berpengaruh terhadap hasil deteksi sarkasme tersebut, terdapat beberapa kendala dalam memilih sebuah dataset yang baik diantaranya adalah :

- Penggunaan bahasa yang informal. Dataset yang diambil dari platform seperti twitter atau X, menggunakan bahasa yang sangat informal sehingga terdapat banyak kosakata yang asing, dan untuk itu belum tersedia *embedding* untuk kata-kata tersebut tidak tersedia [13].
- Diperlukan konteks tambahan untuk memahami sarkasme, contohnya adalah sebuah data yang adalah reply dari komentar sebelumnya, data tersebut tidak dapat dipahami sebagai sarkasme tanpa mengetahui konteks komentar sebelumnya [13].
- Untuk memperoleh dataset yang berkualitas seringkali dibutuhkan pelabelan secara manual, dan itu membutuhkan waktu yang banyak namun hanya menghasilkan data yang sedikit [13].

Oleh karena itu, dalam penelitian kali ini digunakan dataset headline berita, dari website yang didedikasikan untuk membuat berita-berita satir yang mengandung sarkasme yaitu TheOnion, dan headline berita normal yang diambil dari website HuffPost [13]

2.2 BERT

BERT (Bidirectional Encoder Representations from Transformers) adalah model bahasa yang dibangun di atas arsitektur Transformer, dan terdiri dari beberapa lapisan encoder yang masing-masing menggabungkan mekanisme multi-head self-attention dan jaringan saraf untuk memperhitungkan hubungan antara seluruh kata dalam sebuah urutan teks secara kontekstual [14].

Dalam model ini, teks input pertama-tama melalui proses tokenisasi menggunakan WordPiece, yaitu algoritma yang membagi kata menjadi subunit (subwords) untuk mengurangi masalah kata yang tidak terdapat dalam kosakata serta menjaga informasi semantik kata. Tiap token kemudian ditandai dengan token embedding.

Setiap token juga mendapatkan dua jenis embedding tambahan:

- **Segment embedding** berguna menunjukkan apakah token berasal dari kalimat pertama atau kedua dalam input.
- **Position embedding** berguna menyandikan posisi token dalam urutan, karena transformer sendiri tidak memiliki informasi urutan bawaan.

Ketiga embedding tersebut dijumlahkan untuk membentuk representasi vektor akhir setiap token yang kemudian dimasukkan ke dalam encoder transformer untuk menghasilkan representasi kontekstual [14].

2.2.1 Representasi Input/Output model BERT

Dalam penerapan model BERT untuk tugas klasifikasi teks, diperlukan beberapa tahapan prapemrosesan, yaitu tokenisasi, padding, dan encoding. Proses tokenisasi pada BERT dilakukan menggunakan tokenizer bawaan model, di mana teks dipecah menjadi token-token sesuai dengan kosakata BERT. Pada tahap ini, token khusus [CLS] ditambahkan di awal teks sebagai representasi keseluruhan input, sedangkan token [SEP] ditempatkan di akhir teks untuk menandai batas urutan atau pemisah antar kalimat jika terdapat lebih dari satu kalimat dalam input [15].

Tahap berikutnya adalah padding, yang bertujuan untuk menyamakan panjang urutan token pada seluruh data agar sesuai dengan ukuran input yang dibutuhkan oleh model BERT. Sebagai contoh, apabila panjang maksimum token ditetapkan sebanyak 25 token, maka setiap teks yang memiliki jumlah token kurang dari nilai tersebut akan dilengkapi dengan token khusus [PAD] hingga mencapai panjang yang sama. Sebaliknya, teks yang memiliki jumlah token melebihi batas maksimum akan dipotong hingga 25 token, dan token terakhir digunakan sebagai penanda akhir urutan berupa token [SEP] [15].

Tahapan terakhir adalah encoding, yaitu proses mengonversi setiap token menjadi nilai numerik agar dapat diproses oleh model BERT. Pada tahap ini,

token-token yang dihasilkan sebelumnya dipetakan ke dalam bentuk bilangan bulat (integer) berdasarkan kamus token (vocabulary) yang telah disediakan oleh model BERT. Setiap token direpresentasikan oleh integer yang unik, sehingga urutan integer tersebut dapat digunakan oleh model untuk merepresentasikan dan mempelajari informasi dari setiap dokumen secara komputasional [15].

2.2.2 Proses training model BERT

Dalam pengembangan model BERT, proses training dilakukan melalui dua tahapan utama, yaitu pretraining dan fine-tuning. Tahap pretraining merupakan proses awal untuk mempelajari representasi bahasa umum, di mana bobot model yang pada awalnya diinisialisasi secara acak dilatih menggunakan tugas pembelajaran skala besar, seperti Masked Language Modeling (MLM) dan Next Sentence Prediction (NSP), dengan memanfaatkan korpus teks yang sangat besar dan bersifat umum. Melalui tahap ini, model memperoleh pemahaman kontekstual terhadap struktur dan makna bahasa yang kemudian direpresentasikan dalam bentuk bobot pretrained [16].

Tahap selanjutnya adalah fine-tuning, yaitu proses penyesuaian model pretrained untuk menyelesaikan downstream task tertentu, seperti klasifikasi teks, analisis sentimen, atau deteksi sarkasme. Pada tahap ini, sebuah lapisan tambahan dengan jumlah parameter minimal ditambahkan di atas arsitektur BERT, dan seluruh bobot pretrained disesuaikan kembali (updated) menggunakan data berlabel yang relevan dengan tugas spesifik tersebut. Proses fine-tuning memungkinkan model mempertahankan pengetahuan bahasa umum yang telah dipelajari pada tahap pretraining sekaligus mengadaptasikannya secara efektif untuk kebutuhan tugas yang lebih spesifik [16].

2.3 Text Preprocessing

Data atau text preprocessing adalah tahapan yang esensial dalam membangun model machine learning tujuannya adalah untuk membersihkan data dari komponen tertentu yang dapat mempengaruhi hasil deteksi. Pada penelitian ini terdapat beberapa tahap preprocessing.

- **Tokenization** Tokenisasi merupakan tahap awal dalam pemrosesan bahasa alami di mana sebuah teks panjang dibagi menjadi unit-unit yang lebih kecil yang disebut token. Unit ini dapat berupa kata, simbol, atau frasa yang

masing-masing berfungsi sebagai elemen input dalam analisis teks lebih lanjut. Setelah proses ini, data teks yang awalnya berupa paragraf atau kalimat dipecah menjadi daftar token untuk memudahkan pemrosesan dan analisis oleh komputer [17].

- **Case Folding** Dalam pemrosesan teks, case folding adalah proses normalisasi di mana semua huruf dalam dokumen diubah menjadi huruf kecil (lowercase) sehingga variasi huruf besar/kecil tidak menyebabkan istilah yang sama dianggap berbeda. Penelitian terbaru menjelaskan bahwa dengan menyamakan bentuk huruf, tingkat konsistensi teks meningkat dan kesalahan dalam pemetaan kata ke fitur statistik dapat diminimalkan, sehingga tahapan case folding berkontribusi terhadap peningkatan efektivitas analisis tekstual [18].

- **Remove Stopwords** Dalam konteks pemrosesan teks, stopword removal merupakan tahap penting yang bertujuan menghapus kata-kata umum yang sering muncul tetapi tidak memberikan informasi bermakna untuk analisis lebih lanjut, seperti kata penghubung atau kata depan. Penelitian terbaru menunjukkan bahwa meskipun model modern mampu menangani beberapa aspek prapemrosesan, metode klasikal seperti penghapusan stopwords tetap diperlukan untuk mengurangi noise teks dan meningkatkan efektivitas pengolahan data sebelum ekstraksi fitur atau pelatihan model lebih lanjut. Proses ini membantu model fokus pada elemen kata yang menyiratkan makna sehingga performa analisis, seperti klasifikasi teks, cenderung meningkat [19].

2.4 Confusion Matrix

Confusion matrix adalah sebuah tabel yang digunakan untuk mengevaluasi performa model klasifikasi dengan membandingkan hasil prediksi model terhadap label sebenarnya dari data. Struktur tabel ini menunjukkan jumlah prediksi yang benar maupun salah pada masing-masing kelas sehingga dapat memberikan gambaran yang lebih rinci tentang kinerja model dibandingkan sekadar akurasi saja [20]. Berikut tabel confusion matrix

Kelas	Terklasifikasi Positif	Terklasifikasi Negatif
Positif	TP	FP
Negatif	FN	TN

Tabel 2.1. Confusion Matrix

Keterangan :

- True Positive (TP): Jumlah data yang benar-benar positif dan diprediksi positif oleh model
- True Negative (TN): Jumlah data yang benar-benar negatif dan diprediksi negatif oleh model
- False Negative (FN): Jumlah data yang sebenarnya positif tetapi diprediksi negatif
- False Positive (FP): Jumlah data yang sebenarnya negatif tetapi diprediksi positif

Dengan menggunakan empat komponen confusion matrix tersebut, kita dapat menghitung Accuracy, Precicion, Recall, dan F1-Score. Berikut rumus-rumus yang digunakan.

1. Accuracy

   Accuracy (akurasi) adalah metrik yang mengukur proporsi prediksi yang benar dari seluruh prediksi yang dibuat oleh model. Nilai ini mencerminkan seberapa sering model memprediksi dengan tepat baik kelas positif maupun kelas negatif.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

2. Precision

   Precision (presisi) mengukur ketepatan prediksi positif yang dibuat oleh model. Artinya, dari semua instance yang diprediksi sebagai positif, berapa

banyak yang benar-benar positif. Precision penting terutama ketika kesalahan prediksi positif (FP) perlu diminimalkan.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.2}$$

3. Recall

   Recall (sensitivitas) adalah metrik yang menunjukkan kemampuan model dalam menemukan semua instance positif. Dengan kata lain, dari semua data yang sebenarnya positif, berapa banyak yang berhasil diprediksi sebagai positif oleh model. Recall penting ketika kesalahan prediksi negatif (FN) harus diminimalkan.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.3}$$

4. F1-score

   F1 score adalah rata-rata harmonik dari precision dan recall. Metrik ini memberikan ukuran keseimbangan antara precision dan recall, terutama berguna ketika dataset tidak seimbang atau ketika Anda ingin mempertimbangkan kedua metrik tersebut sekaligus. Nilai F1 score berkisar antara 0 hingga 1, di mana nilai mendekati 1 menunjukkan kombinasi precision dan recall yang kuat.

$$F1\text{-Score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.4}$$