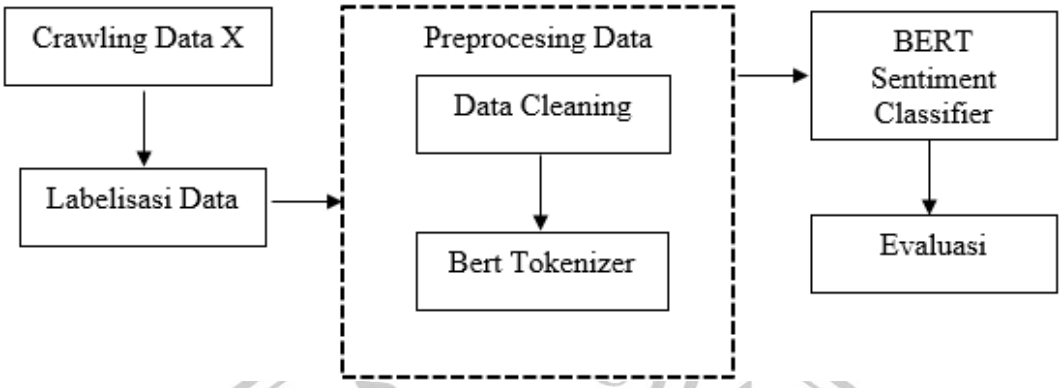
BAB III

METODOLOGI PENELITIAN



Gambar 1. *Flowchart*

3.1 Jenis Data dan Sumber Data

Dalam penelitian ini data yang digunakan merupakan data tabular yaitu jenis data yang disusun dalam bentuk tabel, dengan kolom-kolom yang memiliki atribut atau variabel dan baris-baris yang mewakili entitas atau observasi. Dalam hal ini data tabular yang digunakan berbentuk *file CSV (Comma-Separated Values)* yang diperoleh dengan melakukan data *crawling* pada media sosial X dengan fokus pada data teks yang terkait dengan calon presiden.

3.2 Pengumpulan Data

Pada penelitian ini Data yang akan digunakan adalah data tweet masyarakat indonesia yang berkaitan dengan pemilihan umum presiden 2024. Proses pengumpulan data dilakukan menggunakan teknik *crawling* dengan *library* yang bernama *tweet-harvest* pada media sosial X. *Library* ini bisa mendapatkan semua data tweet yang diinginkan dan hanya bisa diakses menggunakan token akses dari media sosial X. Data Crawling sendiri adalah prosedur pengumpulan data besar yang dapat menjelajah hingga ke halaman web paling dalam [14].

Pengambilan data dilakukan dengan menggunakan *keyword* “pemilu2024” dan “presiden2024” data yang diambil merupakan *tweet* berbahasa indonesia dan diambil pada tanggal 5 Oktober- 21 Oktober 2023.

3.3 Labelisasi Data

Labelisasi data dilakukan secara manual dan dilakukan 3 orang. Proses ini melibatkan pengecekan dan pemahaman data yang ada serta pengklasifikasian data kedalam kategori yang benar. Data dipilih secara acak berdasarkan 3 kategori (positif,netral,dan negatif).

3.4 Preprocessing Data

Setelah pengumpulan data, tahap *preprocessing* dilakukan. Data teks melewati proses pembersihan, termasuk penghapusan karakter khusus dan simbol yang tidak relevan. Data teks kemudian di-tokenisasi menjadi kata-kata yang dapat dianalisis lebih lanjut [15].

1. Data Cleaning

   Data cleaning merupakan proses penghapusan elemen-elemen tidak berguna di dalam sebuah kalimat seperti emoticon, tanda baca, maupun akun yang ada didalam data tweet.

2. *BERT Tokenizer*

   BERT Tokenizer merupakan proses yang dilakukan untuk memecah kalimat menjadi token-token yang lebih kecil seperti kata-kata, sub-kata, atau karakter. Proses implementasi *BERT Tokenizer* menggunakan *library* Hugging Face *Transformer*s[16]. Pada proses klasifikasi menggunakan *BERT* perlu ditambahkannya token khusus yaitu:

   a. [CLS] (*Classification Token*): Token [CLS] ditambahkan di awal teks input untuk menandakan bahwa teks tersebut adalah subjek dari tugas tertentu, seperti klasifikasi teks.

   b. [SEP] (*Separation Token*): Token [SEP] ditambahkan untuk memisahkan segmen-segmen berbeda dalam satu input.

   Data yang telah dibagi menjadi token-token yang lebih kecil dikonversi menjadi ID token sesuai dengan kosakata model *BERT* yang digunakan. Setiap token merupakan ID numerik sesuai dengan yang ada di dalam kosakata.

Setiap kata yang tidak ada di dalam kosakata model akan diwakili dengan token [UNK] (*unknown*) dalam representasi teks. *Attention Mask* akan akan dilakukan untuk menentukan token mana yang akan diberikan perhatian lebih[17]. untuk setiap token yang memiliki nilai lebih dari 0 akan dianggap relevan dan diberikan nilai 1 dan apabila nilainya 0 token tersebut dianggap tidak relevan, maka token tersebut merupakan token [PAD] yang digunakan untuk menyamakan panjang kalimat.

3. Padding

Padding merupakan proses untuk menyamakan panjang dari tiap kalimat sesuai dengan ('MAX_LEN') agar data dapat diproses dengan benar dan konsisten. Untuk kalimat yang lebih sedikit dari ('MAX_LEN') akan ditambahkan dengan [PAD], dan kalimat yang melebihi ('MAX_LEN') akan dipotong untuk menyesuaikan panjang kalimat.

3.5 Data Split

Data yang telah melewati proses Preprocessing akan dibagi menjadi tiga yaitu: data *train*ing , data validasi, dan data *test*. Data validasi diambil dari 15% data *train* selama dilakukannya *train*ing data, data validasi digunakan agar tidak terjadi *overfitting*. Data *test* digunakan sebagai media ukur kinerja model pada pengujian akhir.

Data yang telah dibagi dikonversi menjadi tensor PyTorch sehingga dapat digunakan untuk pelatihan dan pengujian model. Dalam langkah-langkah ini, variabel *train_input, train_labels, train_mask, validation_input, validation_labels, validation_mask, test_input, test_labels,* dan *test_mask* yang telah dipersiapkan sebelumnya dikonversi menjadi *tensor PyTorch* menggunakan *torch.tensor()*. Ini memastikan bahwa data tersebut sekarang berada dalam format yang dapat digunakan dengan model *PyTorch.*

3.6 Pemodelan *BERT*

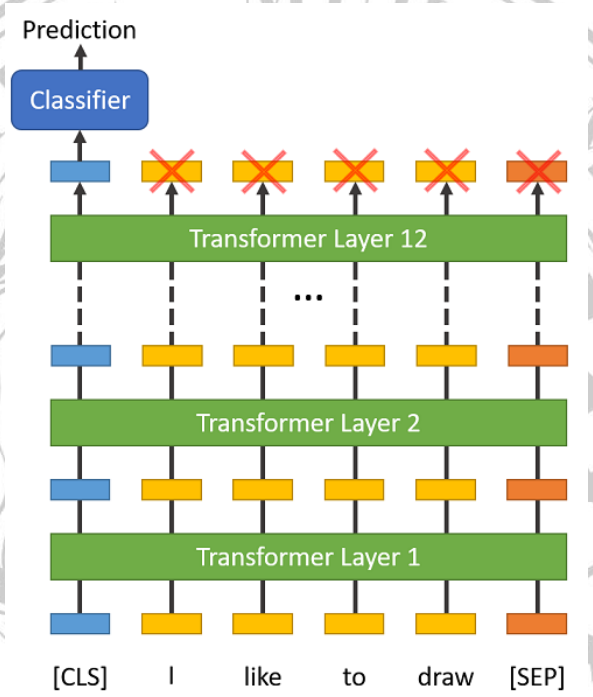
Inisialisasi model *BERT* menggunakan 2 model yang sudah di *pre-trained* yaitu *BERT-base-multilingual-uncased* dan *BERTForSequenceClassification* dengan paramater *num_labels* diatur menjadi 3, untuk mendapatkan tiga kelas keluaran yang berbeda (positif, netral, dan negatif)

1. *BERT-base-multilingual-uncased*

BERT-base-multilingual-uncase merupakan model *BERT* yang telah dilatih dengan berbagai macam bahasa, sehingga memiliki pengetahuan yang bisa digunakan dalam berbagai konteks bahasa.

2. *BERTForSequenceClassification*

BERTForSequenceClassification dirancang khusus untuk mengambil teks atau sekuens kata sebagai input dan menghasilkan prediksi klasifikasi sebagai output. Proses *BERTForSequenceClassification* digambarkan sebagai berikut.



Gambar 2. *Bert Sequence Classification*

Pada gambar 2 menjelaskan bahwa model *BERT* memiliki 12 "pemrosesan" yang disebut lapisan *Transformer*. Ini adalah seperti 12 "pemahaman" berjenjang yang kita berikan pada teks. Hasil dari *BERT Tokenizer* dimasukkan ke lapisan pertama dan menghasilkan keluaran yang telah diperbarui dan menjadi representasi teks yang lebih kaya informasi.

Keluaran dari lapisan pertama diteruskan ke lapisan kedua dan seterusnya hingga lapisan kedua belas. Melalui proses pengulangan ini, representasi teks semakin diperbaiki dan diperkaya menggunakan informasi kontekstual yang lebih baik[18].

Optimizer AdamW akan digunakan untuk mengatur cara model berubah selama pelatihan dengan mengikuti aturan yang telah berikan[12].

Untuk mengatur perubahan tingkat pembelajaran selama pelatihan akan menggunakan fungsi *get_linear_schedule_with_warmup* dengan jumlah *epoch* sebanyak 8 kali. Penjadwalan tingkat pembelajaran akan mengubah tingkat pembelajaran secara *linear* selama pelatihan model selama 8 *epoch*, tanpa langkah pemanasan. Hal ini dapat membantu dalam melatih model dengan cara yang lebih efektif dan mengurangi tingkat data *Loss*.

3.7 Evaluasi

Evaluasi dilakukan menggunakan *confusion matrix*, *confusion matrix* merupakan tabel yang digunakan untuk menggambarkan performa klasifikasi model pada data yang telah dilabeli[19]. Berikut tabel yang digunakan untuk melakukan pengukuran.

Tabel 1. *Confusion Matrix*

Aktual	*Positif*	*Negatif*
Positif	*True Positif (TP)*	*False Positif (FP)*
Negatif	*False Negatif (FN)*	*True Negatif (TN)*

Confusion matrix terdiri dari empat sel utama:

- *True Positive* (TP): Kasus di mana model dengan benar memprediksi kelas positif.
- *True Negative* (TN): Kasus di mana model dengan benar memprediksi kelas negatif.
- *False Positive* (FP): Kasus di mana model salah memprediksi kelas positif (seharusnya negatif).
- *False Negative* (FN): Kasus di mana model salah memprediksi kelas negatif (seharusnya positif).

Confusion matrix digunakan untuk menghitung berbagai metrik evaluasi, seperti accuracy (akurasi), *precision* (presisi), *recall* (sensitivitas), *F1-score*. Berikut rumus *accuracy, precision, recall dan f1-score* :

$$Accuracy = \frac{TP + TN}{TP + TN + FN} + 100$$

$$Precision = \frac{TP}{TP + FP} + 100$$

$$Recall = \frac{TP}{TP + FN} + 100$$

$$F1 - Score = \frac{2 \times Recall \times Precision}{Recall + Precision} + 100$$