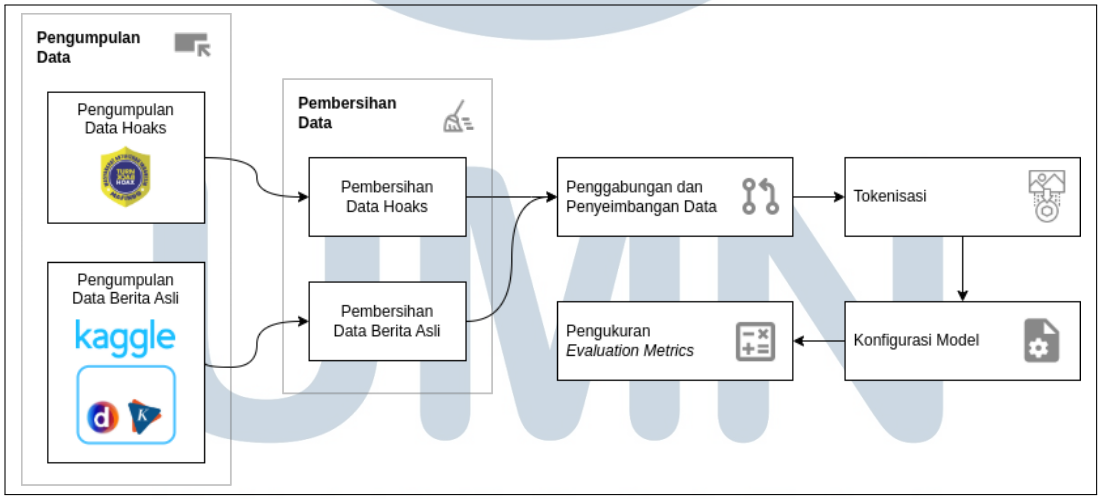
BAB 3
METODOLOGI

Dalam penelitian ini, pendekatan sistematis diterapkan untuk mendeteksi berita hoaks dalam bahasa Indonesia menggunakan model berbasis transformer. Metodologi ini mencakup alur kerja yang komprehensif mulai dari pengumpulan data hingga evaluasi model, yang menangani kompleksitas tugas klasifikasi teks dalam dunia nyata.

Langkah-langkah yang diuraikan dalam bagian ini menjelaskan bagaimana data dikumpulkan, dipra-proses, dan digunakan untuk melakukan fine-tuning model IndoBERT untuk tugas klasifikasi biner, seperti yang diilustrasikan dalam Gambar 3.1. Metodologi ini juga mencakup bagaimana dataset dibagi menjadi kategori berita hoaks dan berita asli, diproses melalui tahapan pembersihan individual, dan digabungkan untuk menciptakan dataset yang seimbang. Proses tokenisasi dan evaluasi model juga diuraikan secara detail untuk memberikan pemahaman lengkap tentang bagaimana model disiapkan untuk deteksi hoaks.



Gambar 3.1. *Flowchart* proses penelitian

3.1 Pengumpulan Data

Penelitian ini menggunakan strategi pengumpulan data multi-sumber untuk memastikan dataset yang kuat dan representatif untuk deteksi hoaks. Oleh karena itu, digunakan dua sumber dataset utama berikut.

3.1.1 Dataset Hoaks

Sumber utama data hoaks adalah API Mafindo, yang disediakan oleh organisasi anti-hoaks Indonesia Mafindo. Pendekatan pengambilan data secara programatik diimplementasikan menggunakan modul `requests` Python untuk mengakses endpoint API pada `https://yudistira.turnbackhoax.id/api/antihoax/` dimana implementasinya dapat dilihat pada Kode 3.1.

```
import requests
import csv

url = "https://yudistira.turnbackhoax.id/api/antihoax/"
batch_size = 1000
total_records = 22712

offset = 0

with open('api_response_data.csv', mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)
    headers_written = False
    while offset < total_records:
        data = {
            'key': '...',
            'limit': batch_size,
            'offset': offset
        }
        response = requests.post(url, data=data)
        if response.status_code == 200:
            json_data = response.json()
            if json_data and isinstance(json_data, list):
                if not headers_written:
                    writer.writerow(json_data[0].keys())
                    headers_written = True

                for entry in json_data:
                    writer.writerow(entry.values())
                offset += batch_size

                print(f"Fetched batch starting at offset {offset}")
            else:
```

```
                print("No more data returned, stopping.")
                break
        else:
            print(f"Failed to fetch data. Status code: {response.
    status_code}, Response: {response.text}")
            break
```

Kode 3.1: Hoax Data Fetching

Proses pengambilan data menghasilkan 14.552 entri, di mana 14.057 entri dilabeli sebagai hoaks dan 495 entri dilabeli sebagai berita asli. Ketidakseimbangan yang signifikan dalam dataset (dimana berita asli hanya sekitar 3% dari total data) mengharuskan pengumpulan sumber berita asli tambahan untuk menciptakan dataset yang lebih seimbang.

3.1.2 Dataset Berita Asli

Untuk melengkapi data berita asli yang terbatas, penelitian ini juga menggunakan *"Indonesia News Dataset (2024)"* dari Kaggle [22]. Dataset tambahan ini menyediakan koleksi artikel berita yang komprehensif dari media massa Indonesia yang faktual; yaitu Kompas.com, Tempo.co dan Detik.com. Dataset ini memiliki sekitar 45.234 artikel yang berkisar dari Januari 2024 hingga September 2024.

Kombinasi data hoaks API Mafindo dan dataset berita asli Kaggle membentuk korpus dasar untuk penelitian deteksi hoaks ini. Sumber yang beragam memastikan representasi yang komprehensif dari konten berita hoaks dan berita sah dalam bahasa Indonesia.

Tabel 3.1. Sumber dan Karakteristik Pengumpulan Data

Sumber Data	**Total Entri**	**Entri Hoaks**	**Entri Berita Asli**
API Mafindo	14.552	14.057	495
Dataset Kaggle	45.234	0	45.234
Total	59.786	14.057	45.729

Proses pengumpulan data dirancang untuk mengatasi potensi bias dan keterbatasan dalam dataset deteksi hoaks yang ada dengan memasukkan berbagai sumber dan memastikan representasi yang komprehensif dari konten berita hoaks dan berita sah.

3.2 Pembersihan Data

Proses pembersihan data merupakan langkah *preprocessing* yang kritis yang dirancang untuk meningkatkan kualitas dan keandalan dataset untuk deteksi hoaks. Proses ini melibatkan teknik normalisasi teks yang canggih yang diterapkan secara terpisah pada dataset hoaks dan berita asli untuk memastikan konsistensi dan menghilangkan noise yang berpotensi menyesatkan model pembelajaran mesin.

3.2.1 Dataset Hoaks

Untuk dataset hoaks yang dikumpulkan dari API Mafindo, proses pembersihan dimulai dengan penyaringan label yang cermat. Banyak entri mengandung metadata atau label status dari API Mafindo yang tidak secara langsung relevan dengan tugas klasifikasi hoaks. Secara khusus, label seperti *"Klarifikasi"*, *"Berita"*, *"Dalam Proses"*, *"Edukasi"*, dan kategorisasi serupa dihapus secara sistematis. Penyaringan selektif ini memastikan bahwa hanya entri hoaks atau berita asli yang jelas yang dipertahankan.

Prosedur pembersihan teks menerapkan beberapa transformasi kunci yang diimplementasikan dalam bentuk Python pada Kode 3.2, dan dijabarkan sebagai berikut.

- Penghapusan karakter khusus dan tanda baca yang dapat memperkenalkan noise ke dalam data tekstual
- Penghilangan URL dan tautan web yang tidak berkontribusi pada konten semantik
- Penghapusan spasi berlebih dan karakter baris baru

```
import pandas as pd
data = pd.read_csv('api_response_data.csv')

df_filtered = data[["status", "classification", "title", "content"]]
df_filtered = df_filtered[df_filtered["classification"] != "CekFakta"]

# List of values to be removed
remove_values = ['Klarifikasi', 'Berita', 'Dalam Proses',
```

```
                'Edukasi', 'Clarification', 'News', 'Education',
    'Sebagian Benar']

# Filter DataFrame to exclude rows where 'status' is in the
    remove_values list
df_filtered = df_filtered[~df_filtered['status'].isin(
    remove_values)]

df_filtered["status"].value_counts()

# Dictionary for mapping status values to 1 or 0
status_mapping = {
    'Salah': 1,
    'False': 1,
    'Belum ada bukti': 1,
    'Benar': 0,
    'True': 0
}

# Apply the mapping to the 'status' column
df_filtered['status_mapped'] = df_filtered['status'].map(
    status_mapping)

# Use str.replace() with a regular expression to remove the label
    at the beginning
df_filtered['title'] = df_filtered['title'].str.replace(r'
    ^\[.*?\]\s*:?|^[A-Z]+\]\s*:?|^[A-Z]+\s*:\s*', '', regex=True)
df_filtered['title'] = df_filtered['title'].str.replace(
    r'^(CEK FAKTA|Cek Fakta|CEK FAKTA Debat|\(CEK FAKTA Debat\))\s
    *[:,\[\]\()*', '', regex=True
)
# These two should replace unicode quotes into regular ASCII
    quotes, but the unicode quotes cannot be shown inside LaTeX
# df_filtered['title'] = df_filtered['title'].str.replace(r
    '["\'"]', '', regex=True)
# df_filtered['content'] = df_filtered['content'].str.replace(r
    '[\"\'""'']', '', regex=True)

df_filtered['content'] = df_filtered['content'].str.replace(r'[\r\
    n]+', ' ', regex=True)

df_filtered = df_filtered.drop(columns=['status', 'classification'
    ])
```

```
df_filtered.rename(columns={'status_mapped': 'is_hoax'}, inplace=
    True)


# remove all missing values
df_filtered = df_filtered.dropna()
df_filtered
df_filtered["is_hoax"].value_counts()

# export df_filtered to csv
df_filtered.to_csv('filtered_data.csv', index=False)
```

Kode 3.2: Hoax Data Cleaning

3.2.2 Dataset Berita Asli

Dataset berita asli menjalani proses pembersihan yang serupa, namun sedikit dimodifikasi. Perhatian khusus diberikan untuk menghapus informasi pengantar situs web dan tag lokasi yang dapat memperkenalkan bias. Misalnya, awalan seperti ”KOMPAS.com” atau indikator lokasi secara sistematis dihilangkan dari awal artikel. Pendekatan ini memastikan bahwa model akan fokus pada konten aktual daripada metadata periferal. Secara garis besar, dilakukan pembersihan teks sebagai berikut.

- Menghapus nama situs web dan informasi publikasi di awal
- Menghilangkan karakter khusus dan tanda baca yang tidak perlu
- Menormalisasi kasus teks dan menghapus spasi berlebih

Langkah-langkah tersebut kemudian diimplementasikan dalam Python dengan menggunakan regex. Adapun kode implementasinya adalah sebagai berikut.

```
def clean_website(text):
    # Check if the input is a string, if not return it as-is (e.g
    ., handle NaN)
    if isinstance(text, str):
        # Check if the text starts with a news website name
    followed by additional information
        if re.match(r'^(KOMPAS.com|TEMPO.CO|detik.com|CNN
    Indonesia|ANTARA News|Reuters|[A-Z\s,]+)', text):
            # Remove everything up to and including the first dash
     character
```

```
            text = re.sub(r'^[A-Z\s,]*(KOMPAS.com|TEMPO.CO|detik.com)[^ -]*[- ]\s*', '', text)
            # Generalized version to remove text before the first dash if it's a location or publisher
            text = re.sub(r'^[A-Z\s,.-]+[- ]\s*', '', text)
        # Remove any leading/trailing whitespace
        text = text.strip()
    return text
```

Kode 3.3: Real News Data Cleaning

3.3 Penggabungan dan Penyeimbangan Data

Penggabungan dan penyeimbangan dataset dilakukan dengan pertimbangan cermat terhadap ketidakseimbangan kelas awal. Dataset Mafindo asli mengandung perbedaan yang signifikan, dengan entri hoaks (14.057) jauh melebihi entri berita asli (495). Untuk mengatasi ketidakseimbangan ini, pendekatan pengambilan sampel acak diterapkan. Dari Indonesia News Dataset yang lebih besar, yang berisi 45.234 artikel berita asli, 15.000 artikel di-*sampling* secara acak dan digabungkan dengan dataset hoaks Mafindo seperti yang tertera pada tabel 3.2. Hanya dua kolom yang akhirnya digunakan untuk pelatihan model, yaitu `content` (Teks lengkap artikel berita) dan `is_hoax` dengan label biner; 0 untuk berita asli, 1 untuk hoaks. Dataset ini kemudian dibagi menjadi subset *train, test,* dan *validation*. Pemisahan ini dilakukan untuk memastikan bahwa setiap subset mempertahankan representasi proporsional artikel hoaks dan berita asli. Dalam Python, pemisahan ini dilakukan dengan menggunakan fungsi `train_test_split` yang terdapat pada *library* `sklearn.model_selection`, seperti yang dijabarkan oleh Kode 3.4.

Tabel 3.2. Jumlah serta Rasio Pembagian Dataset yang digunakan

(a) Jumlah Data Berita

Berita Asli	Berita Hoaks
15.495	14.057
Total: 29.552 artikel	

(b) Rasio Pembagian Data

Training	Test	Validation
80%	10%	10%

```
from sklearn.model_selection import train_test_split
# Split the data into train and test sets (80% train, 10% validation, 10% test)
train_texts, temp_texts, train_labels, temp_labels = train_test_split(
```

```
    news_data['cleaned_content'], labels, test_size=0.2,
    random_state=42, stratify=labels
)

# Further split temp into validation and test sets (50% validation
    , 50% test from remaining 20%)
val_texts, test_texts, val_labels, test_labels = train_test_split(
    temp_texts, temp_labels, test_size=0.5, random_state=42,
    stratify=temp_labels
)
```

Kode 3.4: Train, test, and validation dataset splitting process

3.4 Tokenisasi

Dalam penelitian ini, proses tokenisasi diimplementasikan menggunakan *library HuggingFace Transformers* (`trainer`), khususnya memanfaatkan tokenizer dari model `indobenchmark/indobert-base-p1`, yang dioptimalkan untuk bahasa Indonesia. Proses tokenisasi dari model ini dijabarkan secara algoritmik dengan menggunakan Python pada Kode 3.5.

```
from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained("indobenchmark/indobert-
    base-p1")

def tokenize_text(texts, tokenizer, max_len=512):
    return tokenizer(
        texts.tolist(),
        truncation=True,
        padding=True,
        max_length=max_len,
        return_tensors='np'
    )

# Tokenize the train, validation, and test data
train_encodings = tokenize_text(train_texts, tokenizer)
val_encodings = tokenize_text(val_texts, tokenizer)
test_encodings = tokenize_text(test_texts, tokenizer)
```

Kode 3.5: Tokenizer Function

Di mana dengan menggunakan kode tersebut, proses tokenisasi pada dataset penelitian ini dapat dicontohkan sebagai berikut.

```
Teks Input: "Berita hoax merugikan masyarakat"
Tokenisasi: [CLS] berita hoax merugikan masyarakat [SEP]
Token: [101, 2034, 4231, 3211, 2098, 102]
```

3.5 Konfigurasi Model

Model deteksi hoaks memanfaatkan arsitektur transformer BERT. Model ini kemudian diadaptasi *(fine-tuning)* dalam penelitian ini untuk klasifikasi berita biner.

3.5.1 Arsitektur

Penelitian ini menggunakan model IndoBERT; secara spesifik varian `indobenchmark/indobert-base-p1`, yang menyediakan representasi linguistik bahasa Indonesia yang kuat. Konfigurasi model ini dirancang mengikuti nilai yang direkomendasikan oleh peneliti dari BERT untuk mengoptimalkan kinerja untuk klasifikasi biner, seperti yang diilustrasikan dalam Tabel 3.3.

Tabel 3.3. Parameter Konfigurasi Model

Parameter	**Nilai**
Base Model	IndoBERT
Number of Labels	2
Dropout Probability	0.3
Label Mapping	0: Legitimate News 1: Hoax News

3.5.2 Konfigurasi

Untuk membantu dalam proses konfigurasi *hyperparameter* pada penelitian ini, digunakan library `optuna` untuk melakukan variasi-variasi dalam *range* yang terdapat pada tabel 3.4. Nilai parameter yang digunakan merupakan nilai-nilai yang direkomendasikan oleh Devlin et al. dalam penelitiannya mengenai BERT [13].

Tabel 3.4. Hyperparameter Value

Hyperparameter	**Nilai**
Optimizer	AdamW
Learning Rate	5e−5, 3e−5 dan 2e−5
Weight Decay	1e−2
Ukuran Batch	32
Epoch	2 dan 3

Adapun dalam pengimplementasiannya, dapat diilustrasikan pada fungsi Python yang ada pada Kode 3.6.

```
def objective(trial):
    learning_rate = trial.suggest_categorical('learning_rate', [5e-5, 3e-5, 2e-5])
    epoch = trial.suggest_categorical('epoch', [2, 3, 4])

    (train_encodings, train_labels,
     val_encodings, val_labels,
     test_encodings, test_labels,
     tokenizer) = prepare_data()

    train_dataset = NewsDataset(train_encodings, train_labels)
    val_dataset = NewsDataset(val_encodings, val_labels)

    config = BertConfig.from_pretrained("indobenchmark/indobert-base-p1")
    config.num_labels = 2

    model = BertForSequenceClassification.from_pretrained(
        "indobenchmark/indobert-base-p1",
        config=config
    )

    def compute_metrics(pred):
        labels = pred.label_ids
        preds = pred.predictions.argmax(-1)
        precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='binary')
        acc = accuracy_score(labels, preds)
        return {
            'accuracy': acc,
            'f1': f1,
```

```
            'precision': precision,
            'recall': recall
        }

    training_args = TrainingArguments(
        output_dir='./results',
        num_train_epochs=epoch,
        per_device_train_batch_size=32,
        per_device_eval_batch_size=32,
        learning_rate=learning_rate,
        weight_decay=0.01,
        eval_strategy="epoch",
        save_strategy="no",
        logging_dir='./logs',
    )

    trainer = Trainer(
        model=model,
        args=training_args,
        train_dataset=train_dataset,
        eval_dataset=val_dataset,
        compute_metrics=compute_metrics,
    )

    trainer.train()

    eval_results = trainer.evaluate()

    return eval_results['eval_accuracy']
```

Kode 3.6: Hyperparameter Function

3.6 Pengukuran *Evaluation Metrics*

Pada penelitian ini, proses pengukuran *Evaluation Metrics* dibantu dengan menggunakan library `sklearn.metrics` dimana pengimplementasian library ini dapat dilihat pada Kode 3.7.

```
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    precision, recall, f1, _ = precision_recall_fscore_support(
   labels, preds, average='binary')
```

```
    acc = accuracy_score(labels, preds)
    return {
        'accuracy': acc,
        'f1': f1,
        'precision': precision,
        'recall': recall
    }
```

Kode 3.7: Model Evaluation Process