

Sumber Data : Kaggle “[Locknlock Reviews on Tokopedia](#)”

Dataset ini berisi review pembeli produk lock n lock di Tokopedia. Dataset ini berisi 2 file yaitu TokopediaLocknLock.csv berisi sekitar 10,276 review dan TokopediaLocknLock3k.csv berisi sekitar 3,000 review. Dalam tugas ini digunakan data TokopediaLocknLock.csv yang berisi 10,276 review.

```
df = pd.read_csv("./content/sample_data/TokopediaLocknLock.csv")
df = df.rename(columns={df.columns[0]: "text"})
```

```
print("Jumlah data mentah:", len(df))
df.head()
```

Jumlah data mentah: 10276

	text	grid icon	info icon
0	bahan botol tipis. setauku pet 1 hanya diguna...		
1	Materialnya bagus, tahan lama, ga gampang rus...		
2	packing aman ya dan botol ada plastik pembungk...		
3	kualitas top. model easy grip sangat OK. packi...		
4	Kaget pas datang paketnya dalam dus gede bange...		

Data Preprocessing dan Tokenisasi

Data Preprocessing

```
def clean_text(text):
    text = str(text).lower()                      # lowercase
    text = re.sub(r"http\S+", "", text)           # hapus URL
    text = re.sub(r"@[\w+]", "", text)            # hapus mention
    text = re.sub(r"#\w+", "", text)              # hapus hashtag
    text = re.sub(r"[^a-zA-Z\s]", "", text)        # hapus simbol & angka
    text = re.sub(r"\s+", " ", text).strip()       # hapus spasi berlebih
    return text

df["clean_text"] = df["text"].apply(clean_text)

# Filter Teks Kosong / Terlalu Pendek
df = df[df["clean_text"].str.strip().str.len() > 3].reset_index(drop=True)

print("Jumlah data setelah preprocessing:", len(df))
```

Jumlah data setelah preprocessing: 10260

Dalam pembersihan data, dilakukan mengubah data menjadi lowercase, pembersihan URL, mention, hastag, karakter non-alfabet (simbol & angka) dan spasi berlebih. Hasil ulasan dengan teks terlalu pendek juga dihilangkan. Dari hasil preprocessing tersebut menghasilkan 10.260 data bersih yang bisa digunakan.

```

# Simpan versi asli
df["raw_text"] = df["text"]

# Terapkan preprocessing ke kolom baru
df["clean_text"] = df["text"].astype(str).apply(clean_text)

# Membandingkan sebelum dan sesudah preprocessing
sample = df.sample(5, random_state=42)

for i, row in sample.iterrows():
    print("SEBELUM : ", row["raw_text"])
    print("SESUDAH : ", row["clean_text"])
    print("-" * 80)

SEBELUM : Pengemasan baik. Dus dan botol tidak cacat. Pengiriman cepat. Terima kasih sekali.
SESUDAH : pengemasan baik dus dan botol tidak cacat pengiriman cepat terima kasih sekali
-----
SEBELUM : bgs banget botol ny...mksh banyak tokopedia
SESUDAH : bgs banget botol nymksh banyak tokopedia
-----
SEBELUM : Pengiminan cepat dan packing nya aman. produknya bagus. Bahannya tebal. Sayangnya nggak ada uk besar.
SESUDAH : pengiminan cepat dan packing nya aman produknya bagus bahannya tebal sayangnya nggak ada uk besar
-----
SEBELUM : Sesuai deskripsi pokoknya mantaplah
SESUDAH : sesuai deskripsi pokoknya mantaplah
-----
SEBELUM : produk bagus, tahan panas lama
SESUDAH : produk bagus tahan panas lama
-----
```

Berikut perbandingan data sebelum dan sesudah dilakukan data preprocessing.

```

sentiment_pipe = pipeline(
    "sentiment-analysis",
    models="w1lwo/indonesian-roberta-base-sentiment-classifier",
    device=0 if torch.cuda.is_available() else -1
)

def auto_label(text):
    if text.strip() == "":
        return "neutral"
    result = sentiment_pipe(text[:512])[0]["label"].lower()
    return result

df["label"] = df["clean_text"].apply(auto_label)
print(df["label"].value_counts())

/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret 'HF_TOKEN' does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
warnings.warn(
Device set to use cuda:0
You seem to be using the pipelines sequentially on GPU. In order to maximize efficiency please use a dataset
label
positive     8950
negative     1046
neutral      264
Name: count, dtype: int64
```

Karena data yang ada belum ada pengklasifian label sentimen, proses pelabelan sentimen dilakukan secara otomatis menggunakan model IndoBERT pretrained sebagai pseudo-label yang telah dilatih pada korpus bahasa Indonesia. Hasilnya, model IndoBERT dapat memberikan label positive sebanyak 8950, negative sebanyak 1046 dan neutral sebanyak 264.

```

label_encoder = LabelEncoder()
df["label_id"] = label_encoder.fit_transform(df["label"])

num_labels = len(label_encoder.classes_)
print("Label map:", dict(enumerate(label_encoder.classes_)))

Label map: {0: 'negative', 1: 'neutral', 2: 'positive'}
```

Kemudian setelah mendapat hasilnya, label sentimen yang berbentuk teks akan diubah menjadi nilai numerik agar dapat diproses oleh model machine learning dan deep learning. Model seperti IndoBERT tidak dapat menerima label dalam bentuk string, sehingga setiap kelas sentimen harus direpresentasikan sebagai angka.

```

X_train, X_temp, y_train, y_temp = train_test_split(
    df[["clean_text"]],
    df[["label_id"]],
    test_size=0.30,
    random_state=42,
    stratify=df[["label_id"]]
)

X_val, X_test, y_val, y_test = train_test_split(
    X_temp,
    y_temp,
    test_size=0.50,
    random_state=42,
    stratify=y_temp
)

```

Dari data yang sudah di preprocessing tadi, kemudian dibagi menjadi data training, validation dan testing dengan pembagian 70:15:15. Pembagian ini terlihat dengan test_size = 0.30, artinya data training adalah 70%. Kemudian test_size=0.50 artinya membagi data menjadi 2 untuk data testing dan validasi.

Tokenisasi

```

model_name = "indobenchmark/indobert-base-p1"
tokenizer = AutoTokenizer.from_pretrained(model_name)

train_enc = tokenizer(list(X_train), truncation=True)
val_enc   = tokenizer(list(X_val), truncation=True)
test_enc  = tokenizer(list(X_test), truncation=True)

```

Model yang digunakan adalah indoBERT. Setelah membagi data menjadi data training, validasi dan testing, kemudian dilakukan tahap tokenisasi yang berguna untuk mengubah teks mentah menjadi token ID yang dimengerti oleh model. Proses ini dilakukan karena Model Transformer tidak bisa langsung membaca teks mentah. Hasil tokenisasi akan dipakai untuk PyTorch Dataset dan dimasukkan ke model saat training/evaluasi.

```

class ReviewDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels.tolist()

    def __getitem__(self, idx):
        item = {k: torch.tensor(v[idx]) for k, v in self.encodings.items()}
        item["labels"] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = ReviewDataset(train_enc, y_train)
val_dataset   = ReviewDataset(val_enc, y_val)
test_dataset  = ReviewDataset(test_enc, y_test)

```

Untuk membaca token dan label dari tokenisasi, dataset dibuat. Dalam proses ini dilakukan;

- `__init__` Inisialisasi dataset, dengan menyimpan data encodings (hasil tokenisasi), labels (numerik hasil LabelEncoder) dan memastikan label menjadi list sehingga bisa diakses per index.
- `__getitem__` mengambil satu data untuk training.
- `__len__` untuk memberi tahu PyTorch berapa banyak data dalam dataset agar DataLoader tahu berapa batch yang bisa dibuat.

Kemudian membuat objek dataset yang siap digunakan dengan pembagian train_dataset digunakan untuk fine tuning, val_dataset digunakan untuk evaluasi selama training dan test_dataset digunakan untuk evaluasi akhir.

Handling Imbalance

```
class_weights = compute_class_weight(  
    class_weight="balanced",  
    classes=np.unique(y_train),  
    y=y_train  
)  
class_weights = torch.tensor(class_weights, dtype=torch.float)  
print("Class Weights :", class_weights)  
  
Class Weights : tensor([ 3.2705, 12.9405,  0.3821])
```

Kode ini menghitung bobot untuk setiap kelas agar model tidak bias terhadap kelas yang lebih banyak (imbalanced classes). Proses diterapkan saat training karena memengaruhi cara model dalam memperbaiki error untuk tiap kelas.

class_weight="balanced" artinya dilakukan menghitung bobot otomatis agar kelas minoritas memiliki bobot lebih besar. Proses ini dilakukan untuk memberikan penekanan lebih pada kelas minoritas agar model tidak bias ke kelas yang lebih banyak karena seperti review Tokopedia, jumlah review positif biasanya jauh lebih banyak daripada negatif atau netral.

Fine Tuning

Fine-tuning adalah proses menyesuaikan model yang sudah dilatih sebelumnya (pretrained) dengan dataset atau tugas spesifik yang ingin diselesaikan. Dalam proses ini, menggunakan model indoBert yang sudah pernah dilatih sebelumnya pada dataset besar dan umum.

```
lora_config = LoraConfig(  
    task_type=TaskType.SEQ_CLS,  
    r=8,  
    lora_alpha=32,  
    lora_dropout=0.1  
)
```

Code ini adalah implementasi fine-tuning IndoBERT dengan LoRA (PEFT). LoRA + PEFT dipilih karena memungkinkan fine-tuning IndoBERT secara efisien, hemat memori, cepat, dan tetap efektif untuk dataset ulasan Tokopedia yang relatif kecil. Dalam code ini menyesuaikan layer output dengan jumlah kelas sentimen (positif, netral dan negatif) yang sudah diberi label tadi sebelumnya.

LoRA (Parameter-Efficient Fine-Tuning) adalah teknik fine tuning efisien, yang hanya melatih parameter tambahan kecil daripada semua bobot model. Diketahui rank low-rank matriks LoRA (bobot adaptasi) adalah 8, skala pembobotan LoRA (lora_alpha=32) dan dropout untuk mencegah overfitting pada parameter LoRA adalah 0.1.

Dalam code ini, hanya parameter LoRA yang dilatih sedangkan bobot asli BERT tetap frozen untuk hemat memori dan mempercepat training.

Metrik Evaluasi

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    preds = np.argmax(logits, axis=1)
    return {
        "accuracy": accuracy_score(labels, preds),
        "f1": f1_score(labels, preds, average="weighted")
    }
```

Code ini mendefinisikan fungsi evaluasi untuk digunakan oleh Hugging Face Trainer saat menghitung performa model pada dataset validation atau test. Fungsi ini mengambil logits dan label dari output Trainer, menentukan prediksi kelas (argmax) kemudian menghitung accuracy dan weighted F1 score. F1 score digunakan untuk memperhitungkan jumlah sampel tiap kelas. Jika dataset tidak seimbang, sehingga F1 score minoritas tetap diperhitungkan. Kemudian Trainer akan menampilkan dictionary sebagai hasil evaluasi.

Weighted Trainer

```
class WeightedTrainer(Trainer):
    def compute_loss(self, model, inputs, return_outputs=False, num_items_in_batch=None):
        labels = inputs.pop("labels")
        outputs = model(**inputs)
        logits = outputs.logits
        loss_fn = torch.nn.CrossEntropyLoss(weight=class_weights.to(logits.device))
        loss = loss_fn(logits, labels)
        return (loss, outputs) if return_outputs else loss
```

Code ini digunakan karena pada data sentiment jumlah kelas positif, negatif, netral sering tidak seimbang dan Model cenderung berpihak ke kelas mayoritas. Sehingga Weighted Trainer digunakan untuk meningkatkan performa klasifikasi pada dataset yang tidak seimbang dengan memberikan bobot lebih besar pada kelas minoritas saat training.

Hyperparameter Tuning

Hyperparameter adalah parameter yang ditentukan sebelum training dan tidak dipelajari dari data. Hyperparameter tuning adalah proses mencari kombinasi terbaik dari hyperparameter model agar performanya optimal pada tugas tertentu.

Tujuan Hyperparameter Tuning

- Meningkatkan performa model
- Menyeimbangkan bias dan variance untuk mencegah underfitting/overfitting.
- Memaksimalkan efisiensi untuk training lebih cepat, penggunaan memori optimal.

```
experiment_configs = [
    {"epochs": 5, "lr": 2e-5, "batch_size": 16},
    {"epochs": 5, "lr": 2e-5, "batch_size": 32},
    {"epochs": 10, "lr": 2e-5, "batch_size": 16},
    {"epochs": 10, "lr": 2e-5, "batch_size": 32},
    {"epochs": 5, "lr": 3e-5, "batch_size": 16},
    {"epochs": 5, "lr": 3e-5, "batch_size": 32},
    {"epochs": 10, "lr": 3e-5, "batch_size": 16},
    {"epochs": 10, "lr": 3e-5, "batch_size": 32}
]
```

Learning Rate

Pada eksperimen ini digunakan learning rate 2e-5 dan 3e-5. Learning rate 2e-5 dan 3e-5 dipilih karena merupakan rentang yang umum digunakan pada fine-tuning model BERT dan mampu menjaga stabilitas proses pelatihan. Rentang ini cukup sempit sehingga perubahan performa dapat diamati secara jelas.

Batch Size

Batch size yang digunakan pada eksperimen adalah 16 dan 32. Batch size 16 dipilih karena memberikan gradien yang lebih stabil dan sesuai dengan keterbatasan memori GPU pada Google Colab sedangkan Batch size 32 digunakan untuk membandingkan pengaruh batch yang lebih besar terhadap kecepatan dan stabilitas training. Kedua nilai ini merupakan konfigurasi standar dalam fine tuning Transformer untuk dataset berukuran seperti 10.000ribuan data.

Epoch

Jumlah epoch yang digunakan dalam eksperimen adalah 5 dan 10 epoch. Epoch 5 digunakan untuk melihat performa model dengan pelatihan minimal dan Epoch 10 digunakan untuk mengamati potensi peningkatan performa serta risiko overfitting. Variasi ini memungkinkan analisis pengaruh durasi training terhadap performa model.

Berikut adalah hasil dari eksperimen yang telah dilakukan dengan menggunakan Learning Rate: 2e-5 dan 3e-5, Batch Size: 16 dan 32 dan Epoch: 5 dan 10.

```
== Experiment 1 | Epoch 5 | LR 2e-05 | Batch 16 ==
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[2245/2245 02:56, Epoch 5/5]
```

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.990000	0.905803	0.779077	0.806107
2	0.667600	0.775552	0.890188	0.889299
3	0.833500	0.747210	0.913580	0.907738
4	0.740800	0.742970	0.917479	0.910044
5	0.663700	0.728596	0.913580	0.907993

```
== Experiment 2 | Epoch 5 | LR 2e-05 | Batch 32 ==
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[1125/1125 02:52, Epoch 5/5]
```

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	1.025700	0.993018	0.749838	0.783364
2	0.901800	0.819505	0.825861	0.840812
3	0.802700	0.722386	0.855101	0.864247
4	0.737000	0.691718	0.873944	0.879137
5	0.652300	0.679819	0.866797	0.874144

==== Experiment 3 | Epoch 10 | LR 2e-05 | Batch 16 ====
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[4490/4490 05:43, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.925000	0.832168	0.768681	0.799217
2	0.627600	0.714350	0.901235	0.897983
3	0.751300	0.681568	0.913580	0.907359
4	0.680200	0.676395	0.927875	0.919243
5	0.596400	0.683782	0.925276	0.917278
6	0.556700	0.659135	0.920078	0.913460
7	0.556400	0.633808	0.920078	0.915589
8	0.581300	0.630675	0.922027	0.918700
9	0.540200	0.634426	0.920728	0.917795
10	0.531600	0.641244	0.919428	0.916131

==== Experiment 4 | Epoch 10 | LR 2e-05 | Batch 32 ====
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[2250/2250 05:45, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	1.078800	1.041890	0.688109	0.735377
2	0.954300	0.832866	0.847303	0.856689
3	0.742700	0.686813	0.891488	0.895910
4	0.678100	0.631581	0.879792	0.891193
5	0.569600	0.608083	0.888889	0.901196
6	0.565700	0.584402	0.873294	0.891517
7	0.545200	0.574819	0.879142	0.896348
8	0.530800	0.576541	0.882391	0.897677
9	0.499000	0.562528	0.878493	0.896500
10	0.496700	0.561138	0.878493	0.896357

==== Experiment 5 | Epoch 5 | LR 3e-05 | Batch 16 ====
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[2245/2245 02:53, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.866500	0.731810	0.835608	0.849834
2	0.608000	0.674915	0.899935	0.897212
3	0.740600	0.674720	0.918129	0.910332
4	0.656900	0.680688	0.923977	0.917079
5	0.571600	0.644968	0.918778	0.914692

==== Experiment 6 | Epoch 5 | LR 3e-05 | Batch 32 ====
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[1125/1125 02:53, Epoch 5/5]

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.986900	0.901157	0.775179	0.804628
2	0.765900	0.700776	0.840806	0.853939
3	0.696600	0.649705	0.888889	0.893978
4	0.651300	0.630836	0.895387	0.902564
5	0.568900	0.615392	0.886940	0.896671

```
== Experiment 7 | Epoch 10 | LR 3e-05 | Batch 16 ==
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[4490/4490 05:45, Epoch 10/10]
```

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	0.818200	0.735363	0.813515	0.832972
2	0.601000	0.687136	0.913580	0.907877
3	0.710700	0.666431	0.921378	0.914528
4	0.645400	0.640379	0.930474	0.925134
5	0.560400	0.664185	0.928525	0.923717
6	0.488900	0.630727	0.923977	0.921976
7	0.486500	0.598178	0.922677	0.923107
8	0.514400	0.594421	0.922677	0.923367
9	0.459700	0.591153	0.922677	0.924275
10	0.453800	0.601284	0.923977	0.924691

```
== Experiment 8 | Epoch 10 | LR 3e-05 | Batch 32 ==
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at indobenchmark/indobert-base-p1 and are newly initialized.
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
[2250/2250 05:47, Epoch 10/10]
```

Epoch	Training Loss	Validation Loss	Accuracy	F1
1	1.048700	0.975942	0.712801	0.757605
2	0.759600	0.686469	0.859649	0.870804
3	0.623400	0.607575	0.895387	0.905402
4	0.580400	0.545276	0.873294	0.893906
5	0.486100	0.537105	0.881741	0.898966
6	0.462100	0.545216	0.884990	0.900076
7	0.465300	0.544217	0.886290	0.900765
8	0.448100	0.561511	0.892138	0.904718
9	0.416100	0.538797	0.886940	0.901043
10	0.421200	0.540708	0.888239	0.901781

[49/49 00:03]

Setelah Eksperimen dilakukan, kemudian dilakukan mengumpulkan hasil evaluasi dengan menampilkan nilai accuracy dan F1 score dari setiap kombinasi epoch, learning rate dan batch size.

Kemudian dari eksperimen diatas menghasilkan ;

experiment	epochs	learning_rate	batch_size	accuracy	f1
6	7	10	0.00003	16	0.923977 0.924691
2	3	10	0.00002	16	0.919428 0.916131
4	5	5	0.00003	16	0.918778 0.914692
0	1	5	0.00002	16	0.913580 0.907993
7	8	10	0.00003	32	0.888239 0.901781
5	6	5	0.00003	32	0.886940 0.896671
3	4	10	0.00002	32	0.878493 0.896357
1	2	5	0.00002	32	0.866797 0.874144

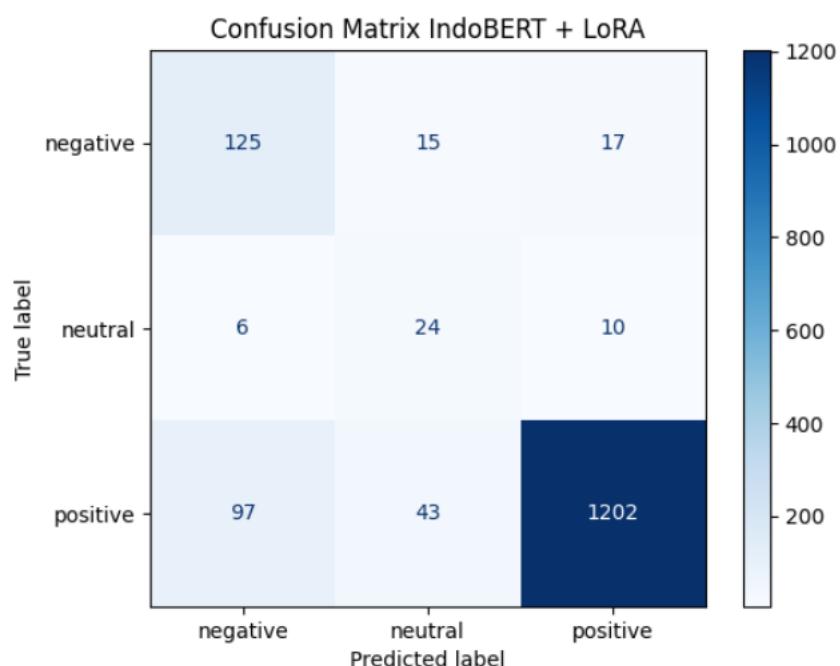
Berdasarkan hasil eksperimen yang dilakukan dengan variasi epoch, learning rate dan batch size, dapat disimpulkan bahwa konfigurasi hyperparameter berpengaruh signifikan terhadap performa model IndoBERT + LoRA dalam klasifikasi sentimen.

Konfigurasi terbaik diperoleh pada:

- Epoch = 10
- Learning rate = 3e-5
- Batch size = 16

Konfigurasi tersebut menghasilkan nilai accuracy sebesar 0.9239 dan F1-score sebesar 0.9247, yang merupakan nilai tertinggi dibandingkan konfigurasi lainnya.

Confusion Matrix



Berdasarkan confusion matrix diatas, dapat disimpulkan bahwa model IndoBERT + LoRA mampu mengklasifikasikan sentimen dengan sangat baik pada kelas positif, namun masih mengalami tantangan pada pembedaan kelas negatif dan netral.

Terdapat 1202 data positif berhasil diklasifikasikan dengan benar dan merupakan nilai tertinggi pada confusion matrix. Sehingga dapat disimpulkan Model sangat efektif dalam mengenali sentimen positif, menunjukkan bahwa pola bahasa positif pada data cukup kuat dan konsisten. Kesalahan klasifikasi paling sering terjadi pada kelas netral, yang sering tertukar dengan positif atau negative seperti negatif (6) dan positif (10). Ketidakseimbangan data berpengaruh terhadap performa klasifikasi, meskipun telah diterapkan class weighting.

