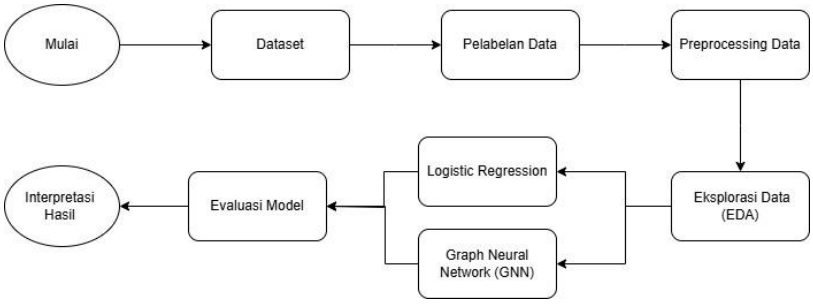
MUHAMMAD RIZKI NURFIQRI
TASK KANDIDAT DATA SCIENTIST

Laporan Dokumentasi Analisis Sentimen Twitter

Laporan ini mendokumentasikan analisis sentimen data Twitter, memanfaatkan teknologi seperti *pretrained* model IndoBERT untuk pelabelan data dan penerapan algoritma *Logistic Regression* serta *Graph Neural Network* (GNN) untuk klasifikasi. Pemilihan topik ini relevan dengan kebutuhan industri untuk memahami opini publik yang berkembang melalui media sosial, terutama Twitter, yang sering menjadi platform diskusi paling dinamis. Berikut adalah workflow yang digunkana untuk penelitian ini



Workflow analisis sentimen Twitter dimulai dari pengumpulan data, di mana data mentah diambil dari sebuah dataset yang tersedia di situs Github dalam format CSV. Dataset ini sebenarnya sudah memiliki label untuk hate speech dan abusive language, namun label tersebut tidak digunakan karena fokus analisis adalah pada klasifikasi sentimen (positif, negatif, netral), bukan pada deteksi ujaran kebencian. Oleh karena itu, langkah berikutnya adalah pelabelan otomatis menggunakan IndoBERT untuk menentukan sentimen (positif, negatif, netral), karena data asli tidak memiliki label. Setelah itu, data melalui tahap preprocessing, termasuk pembersihan teks, tokenisasi, penghapusan stop words, dan representasi numerik menggunakan TF-IDF.

Setelah pelabelan, data melalui tahap **preprocessing**, seperti pembersihan teks, tokenisasi, penghapusan stop words, dan representasi numerik menggunakan TF-IDF. Dilanjutkan dengan **eksplorasi data**, dilakukan analisis distribusi sentimen, panjang tweet, dan kata-kata yang paling umum untuk memahami pola dalam data. Pada tahap **pengembangan model**, dua pendekatan diuji: Logistic Regression, yang memanfaatkan TF-IDF, dan Graph Neural Network (GNN), yang mengeksplorasi hubungan antar-node. Logistic Regression menunjukkan performa lebih baik dengan akurasi 70.24% dan F1-score 68.53%, dibandingkan dengan GNN yang mencapai akurasi 60.93% dan F1-score 60.88%. **Evaluasi model** menegaskan bahwa Logistic Regression lebih cocok untuk data sederhana ini. Akhirnya, tahap **interpretasi hasil** menyimpulkan bahwa Logistic Regression lebih efisien dan efektif untuk analisis sentimen teks berbahasa Indonesia, dengan fokus yang jelas pada klasifikasi sentimen.

MUHAMMAD RIZKI NURFIQRI
TASK KANDIDAT DATA SCIENTIST

Alasan Memilih Topik Analisis Sentimen

1. **Relevansi Strategis**: Analisis sentimen membantu perusahaan, organisasi, dan lembaga pemerintahan dalam memahami opini masyarakat terhadap produk, layanan, atau kebijakan tertentu.
2. **Kekuatan Media Sosial**: Twitter, dengan format pesannya yang singkat, menjadi salah satu sumber utama data untuk memahami opini publik. Topik ini mengeksplorasi bagaimana analisis sentimen dapat memberikan wawasan praktis.
3. **Teknologi Berbasis Bahasa Lokal**: Pemanfaatan model seperti IndoBERT menunjukkan bagaimana solusi berbasis AI dapat disesuaikan untuk data lokal, dalam hal ini bahasa Indonesia.

1. Pengumpulan Data

Data Twitter diambil dari situs Github dalam format CSV. Dataset ini sebenarnya sudah memiliki label untuk *hate speech* dan *abusive language*, tetapi tidak digunakan karena fokus analisis adalah pada klasifikasi sentimen (positif, negatif, atau netral). Oleh karena itu, data dianggap mentah untuk keperluan pelabelan sentimen.

Alasan

- **Fokus pada Sentimen**: Dataset asli berfokus pada *hate speech* dan *abusive language*, yang tidak relevan dengan tujuan analisis sentimen ini.
- **Fleksibilitas**: Dengan melabeli ulang data menggunakan IndoBERT, analisis dapat disesuaikan dengan kebutuhan klasifikasi sentimen secara spesifik.

Keuntungan

- **Efisiensi**: Proses pelabelan otomatis menghemat waktu dibandingkan pelabelan manual.
- **Akurasi**: Menggunakan model pretrained seperti IndoBERT memberikan baseline label yang cukup akurat, khususnya untuk data teks berbahasa Indonesia.

MUHAMMAD RIZKI NURFIQRI
TASK KANDIDAT DATA SCIENTIST

2. Pelabelan Data Menggunakan IndoBERT

IndoBERT (bagian dari IndoBench) digunakan untuk memberi label sentimen pada data mentah. IndoBERT adalah model bahasa berbasis BERT yang terlatih pada data teks bahasa Indonesia, termasuk tugas analisis sentimen.

Kode untuk Melabeli Data Menggunakan IndoBERT

```python
import pandas as pd
from transformers import BertTokenizer, BertForSequenceClassification
import torch

# Muat model dan tokenizer IndoBERT
tokenizer = BertTokenizer.from_pretrained("indobenchmark/indobert-base-p2")
model = BertForSequenceClassification.from_pretrained("indobenchmark/indobert-base-p2")

# Fungsi untuk melakukan prediksi sentimen
def analyze_sentiment(text):
    # Pastikan input adalah string dan tidak kosong
    if isinstance(text, str) and text.strip():
        try:
            # Tokenisasi input text
            inputs = tokenizer(text, return_tensors="pt", padding=True, truncation=True, max_length=512)

            # Debug: Cek hasil tokenisasi
            print(f"Tokenized input: {inputs}")

            # Prediksi dengan model
            with torch.no_grad():
                outputs = model(**inputs)

            # Periksa apakah outputs memiliki data logits
            if hasattr(outputs, 'logits'):
                logits = outputs.logits
                prob = torch.nn.functional.softmax(logits, dim=-1)

                # Ambil label prediksi dengan probabilitas tertinggi
                sentiment_labels = ["negative", "neutral", "positive"]

                # Pastikan prediksi berada dalam rentang yang valid
                predicted_class = torch.argmax(prob, dim=-1).item()

                # Output prediksi
                return sentiment_labels[predicted_class], prob[0][predicted_class].item()
            else:
                print("Error: No logits found in the model output.")
                return "Error", 0.0
        except Exception as e:
            # Jika ada error lain dalam pemrosesan, tangkap dan cetak
            print(f"Error processing text: {text} -> {e}")
            return "Error", 0.0
    else:
        print(f"Skipping invalid input: {text}")
        return "Error", 0.0

# Pastikan kolom 'Tweet' dalam tipe data string
df['Tweet'] = df['Tweet'].astype(str)

# Cek beberapa contoh tweet di dataset
print(df.head())

# Prediksi sentimen untuk setiap baris dalam DataFrame
df['sentiment'], df['confidence'] = zip(*df['Tweet'].apply(analyze_sentiment))

# Menampilkan hasil pada beberapa baris pertama
print(df[['Tweet', 'sentiment', 'confidence']].head())

# Simpan hasil analisis sentimen ke file baru (opsional)
df.to_csv("dataset_with_sentiment.csv", index=False)
```

```python
# Mengganti nilai 'Error' menjadi 'neutral' di kolom 'sentiment'
df['sentiment'] = df['sentiment'].replace('Error', 'neutral')

# Menampilkan beberapa baris pertama untuk memverifikasi
print(df[['Tweet', 'sentiment', 'confidence']].head())

# Menyimpan hasil yang sudah diperbarui ke file baru (opsional)
df.to_csv("dataset_with_sentiment_updated.csv", index=False)
```

Alasan

- **Alasan Menggunakan IndoBERT**: IndoBERT telah dilatih pada teks bahasa Indonesia dan mencakup tugas analisis sentimen, sehingga cocok untuk memproses tweet dalam bahasa Indonesia.
- **Keuntungan**: IndoBERT memungkinkan pelabelan otomatis yang relatif akurat, menghindari proses labeling manual dan memberikan dasar yang baik untuk tahap berikutnya.

2. Preprocessing Data

Setelah data dilabeli, preprocessing dilakukan dengan langkah-langkah berikut:

- **Membersihkan teks**: Menghapus URL, tanda baca, dan karakter spesial.
- **Tokenisasi dan Penghapusan Stop Words**: Memecah teks menjadi kata-kata dan menghapus kata-kata umum.
- **TF-IDF Vectorization**: Mengubah teks menjadi representasi numerik menggunakan TF-IDF.

Kode Preprocessing (Sesuai Notebook)

```
# Preprocessing Data
def clean_text(text):
    text = re.sub(r'http\S+', '', text)  # Menghapus URL
    text = re.sub(r'@\w+', '', text)     # Menghapus mention
    text = re.sub(r'#[\w-]+', '', text)  # Menghapus hashtag
    text = re.sub(r'[^\w\s]', '', text)  # Menghapus tanda baca
    text = re.sub(r'USER', '', text)  # Menghapus tanda baca
    text = re.sub(r'URL', '', text)  # Menghapus tanda baca
    text = text.lower().strip()          # Lowercase dan strip spasi
    return text

# Terapkan fungsi clean_text ke kolom Tweet
df['Cleaned_Tweet'] = df['Tweet'].apply(clean_text)

# Label encoding untuk kolom sentimen
le = LabelEncoder()
df['sentiment_label'] = le.fit_transform(df['sentiment'])
```

Alasan

- **Menggunakan TF-IDF**: TF-IDF sangat efektif untuk representasi numerik dalam analisis sentimen teks.
- **Keuntungan**: Dengan representasi yang terstruktur, model machine learning dapat beroperasi lebih efisien.

3. Eksplorasi Data

Pada tahap eksplorasi, kami melakukan analisis distribusi label sentimen, mencari kata yang paling umum untuk setiap label, dan memahami Panjang tweet

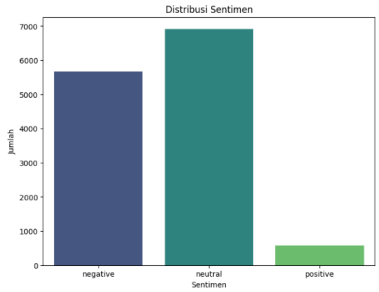
Alasan

- **Alasan Analisis Distribusi**: Mengetahui distribusi ini membantu dalam menyesuaikan strategi pengambilan sampel jika data tidak seimbang.
- **Keuntungan**: Eksplorasi ini memungkinkan kita untuk memahami karakteristik data dan persiapan data yang lebih baik untuk model.

MUHAMMAD RIZKI NURFIQRI
TASK KANDIDAT DATA SCIENTIST

A. Distribusi Label Sentimen

Dataset ini memiliki distribusi label sentimen sebagai berikut:

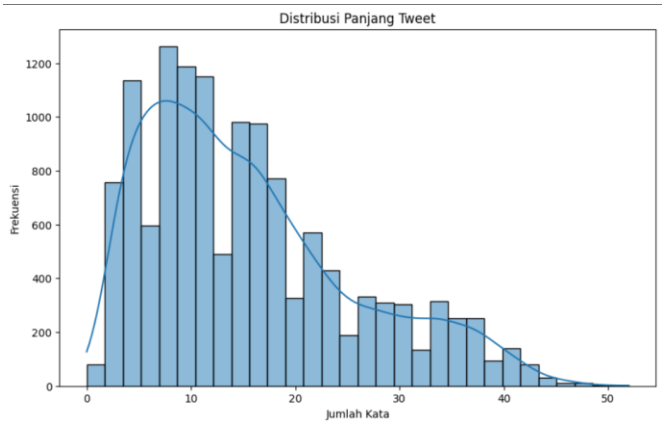


Neutral	6917 sampel
Negative	5669 sampel
Positive	583 sampel

Jumlah sampel yang jauh lebih rendah pada label **positif** menunjukkan ketidakseimbangan data, yang penting diperhatikan dalam pemilihan dan evaluasi model.

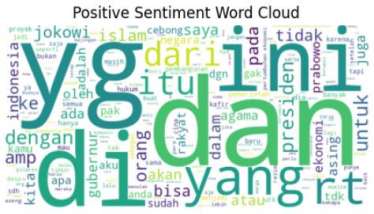
B. Distribusi Panjang Tweet

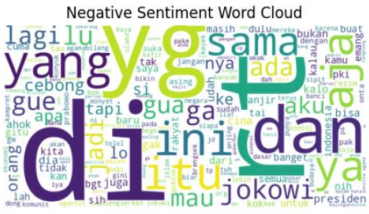
Visualisasi distribusi panjang tweet ini membantu memahami rata-rata panjang tweet dalam dataset, yang penting untuk menentukan pemrosesan teks lebih lanjut atau penyesuaian model.

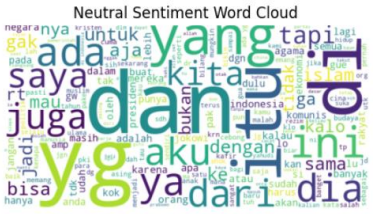


C. Visualisasi Kata yang Paling Sering Muncul pada Setiap Label

Visualisasi ini menunjukkan kata-kata yang paling sering muncul pada tweet dengan label **positif**, **negatif**, dan **netral**, yang memberikan insight tambahan tentang topik atau nada yang paling umum dalam tiap kategori sentimen.







Hasil Eksplorasi

1. **Distribusi Label Sentimen**: Distribusi yang terlihat tidak seimbang, dengan kategori **positif** yang jauh lebih sedikit, yang berpotensi mempengaruhi performa model. Teknik seperti penanganan ketidakseimbangan data (resampling atau penyesuaian parameter model) dapat dipertimbangkan.
2. **Distribusi Panjang Tweet dan Word Cloud**: Sama seperti sebelumnya, analisis ini membantu kita dalam menentukan pendekatan pemodelan yang sesuai.

4. Pengembangan Model

A. Logistic Regression (Notebook: Logistic_Regrecion.ipynb)

Logistic Regression dipilih sebagai baseline model karena sederhana dan efektif untuk data teks yang diproses dengan TF-IDF. Model ini melatih klasifikasi sentimen dan menghasilkan evaluasi metrik seperti akurasi, precision, recall, dan F1-score.

Kode Logistic Regression

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

# Inisialisasi LabelEncoder
le = LabelEncoder()

# Label encoding untuk kolom sentimen
df['sentiment_label'] = le.fit_transform(df['sentiment'])

# Pastikan semua data Tweet adalah string
df['Tweet'] = df['Cleaned_Tweet'].fillna('')

# Split data menjadi training dan testing
X = df['Tweet']  # Menggunakan kolom 'Tweet' sebagai fitur
y = df['sentiment_label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Pastikan X_train dan X_test berisi teks yang valid
assert X_train.apply(lambda x: isinstance(x, str)).all(), "X_train contains non-string values!"
assert X_test.apply(lambda x: isinstance(x, str)).all(), "X_test contains non-string values!"

# Menggunakan TF-IDF untuk vektorisasi hanya pada teks (Tweet)
tfidf = TfidfVectorizer(max_features=5000)

# Vektorisasi data
X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf = tfidf.transform(X_test)

# Cek hasil vektorisasi
print(X_train_tfidf.shape)

# Model Logistic Regression
model = LogisticRegression(max_iter=1000)

# Latih model dengan data yang sudah tervektorisasi
model.fit(X_train_tfidf, y_train)

# Prediksi dan evaluasi
y_pred = model.predict(X_test_tfidf)
```

```python
# Evaluasi model
accuracy = accuracy_score(y_test, y_pred) * 100
precision = precision_score(y_test, y_pred, average='weighted') * 100
recall = recall_score(y_test, y_pred, average='weighted') * 100
f1 = f1_score(y_test, y_pred, average='weighted') * 100

# Menampilkan hasil dalam persen
print(f"Accuracy: {accuracy:.2f}%")
print(f"Precision: {precision:.2f}%")
print(f"Recall: {recall:.2f}%")
print(f"F1 Score: {f1:.2f}%")
```

```python
# Menampilkan classification report
print(classification_report(y_test, y_pred, target_names=le.classes_))
```

Alasan

- **Alasan Penggunaan Logistic Regression**: Logistic Regression memberikan baseline yang sederhana untuk mengukur performa klasifikasi teks tanpa memerlukan model neural network.
- **Keuntungan**: Cepat dan efektif, terutama dengan data terstruktur sederhana yang diwakili dengan TF-IDF.

MUHAMMAD RIZKI NURFIQRI
TASK KANDIDAT DATA SCIENTIST

B. Graph Neural Network (GNN) (Notebook: GNN.ipynb)

Graph Neural Network (GNN) diimplementasikan untuk memanfaatkan hubungan antara node jika data memiliki informasi graf. GNN biasanya cocok untuk data yang memiliki konektivitas, namun dalam studi ini dicoba sebagai eksperimen untuk membandingkan performa.

Kode Graph Neural Network

```python
# Label encoding untuk kolom sentimen
le = LabelEncoder()
df['sentiment_label'] = le.fit_transform(df['sentiment'])

# Pastikan semua data Tweet adalah string
df['Tweet'] = df['Cleaned_Tweet'].fillna("")

# Split data menjadi training dan testing
X = df['Tweet']
y = df['sentiment_label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Menggunakan TF-IDF untuk vektorisasi hanya pada teks (Tweet)
tfidf = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf.fit_transform(X)

# Konversi ke array NumPy
X_features = X_tfidf.toarray()

# Membuat adjacency matrix berdasarkan cosine similarity
from sklearn.metrics.pairwise import cosine_similarity
similarity_matrix = cosine_similarity(X_features)
```

```python
# Threshold untuk adjacency matrix (hanya koneksi dengan similarity > threshold)
threshold = 0.5
adjacency_matrix = (similarity_matrix > threshold).astype(int)

# Konversi adjacency matrix menjadi edge index untuk PyTorch Geometric
edge_index = np.array(adjacency_matrix.nonzero(), dtype=np.int64)

# Membuat data PyTorch Geometric
data = Data(
    x=torch.tensor(X_features, dtype=torch.float),
    edge_index=torch.tensor(edge_index, dtype=torch.long),
    y=torch.tensor(y, dtype=torch.long)
)

# Split training dan testing berdasarkan indeks
train_mask = torch.zeros(data.num_nodes, dtype=torch.bool)
test_mask = torch.zeros(data.num_nodes, dtype=torch.bool)

train_indices = X_train.index
test_indices = X_test.index

train_mask[train_indices] = True
test_mask[test_indices] = True

data.train_mask = train_mask
data.test_mask = test_mask
```

```python
# Definisikan model GCN
class GCN(torch.nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(GCN, self).__init__()
        self.conv1 = GCNConv(input_dim, hidden_dim)
        self.conv2 = GCNConv(hidden_dim, output_dim)

    def forward(self, data):
        x, edge_index = data.x, data.edge_index
        x = self.conv1(x, edge_index)
        x = F.relu(x)
        x = self.conv2(x, edge_index)
        return F.log_softmax(x, dim=1)

# Inisialisasi model, optimizer, dan loss function
model = GCN(input_dim=5000, hidden_dim=64, output_dim=len(le.classes_))
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
criterion = torch.nn.CrossEntropyLoss()

# Training GNN
def train(data):
    model.train()
    optimizer.zero_grad()
    out = model(data)
    loss = criterion(out[data.train_mask], data.y[data.train_mask])
    loss.backward()
    optimizer.step()
    return loss.item()
```

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Fungsi untuk mendapatkan prediksi dan label asli
def evaluate_predictions(data):
    model.eval()
    with torch.no_grad():
        logits = model(data)
        pred = logits[data.test_mask].argmax(dim=1)
        true_labels = data.y[data.test_mask]

        # Konversi ke NumPy
        pred_np = pred.cpu().numpy()
        true_labels_np = true_labels.cpu().numpy()

        return pred_np, true_labels_np

# Loop training
for epoch in range(1, 201):  # 200 epochs
    loss = train(data)  # Fungsi train melatih model
    if epoch % 10 == 0:  # Cetak setiap 10 epoch
        print(f"Epoch: {epoch:03d}, Loss: {loss:.4f}")

# Setelah pelatihan selesai
print("Final Evaluation on Test Data:")

# Dapatkan prediksi dan label asli
pred_np, true_labels_np = evaluate_predictions(data)

# Hitung metrik
accuracy = accuracy_score(true_labels_np, pred_np) * 100
precision = precision_score(true_labels_np, pred_np, average='weighted') * 100
recall = recall_score(true_labels_np, pred_np, average='weighted') * 100
f1 = f1_score(true_labels_np, pred_np, average='weighted') * 100

# Cetak hasil evaluasi
print(f"Accuracy: {accuracy:.2f}%")
print(f"Precision: {precision:.2f}%")
print(f"Recall: {recall:.2f}%")
print(f"F1 Score: {f1:.2f}%")
```

Alasan

- **Alasan Penggunaan GNN**: Meskipun GNN lebih cocok untuk data yang memiliki koneksi graf, dicoba di sini untuk melihat apakah ada peningkatan performa dibandingkan Logistic Regression.
- **Keuntungan**: GNN mampu memproses data dengan pola hubungan antar-node, walaupun pada dataset ini hasilnya mungkin tidak optimal.

5. Evaluasi Model

Kedua model dievaluasi dengan metrik akurasi, precision, recall, dan F1-score. Hasil yang diperoleh adalah:

- **Logistic Regression**:

Accuracy	**70.24%**
Precision	**66.96%**
Recall	**70.24%**
F1 Score	**68.53%**

- **Graph Neural Network (GNN)**:

Accuracy	60.93%
Precision	60.85%
Recall	60.93%
F1 Score	60.88%

Analisis Hasil:

- Logistic Regression unggul karena representasi TF-IDF lebih sesuai dengan model sederhana berbasis statistik.
- GNN tidak menunjukkan peningkatan performa signifikan karena data tidak memiliki hubungan graf yang cukup kuat**.**

Kesimpulan

Laporan ini menunjukkan pentingnya memilih model yang sesuai dengan karakteristik data. sehingga setelah dilakukan nya analisis sentimen ini, dapat diketahui bahwa Logistic Regression, dengan kesederhanaan dan efisiensinya, memberikan hasil yang lebih baik dibandingkan GNN. Ini dapat disebabkan oleh data yang diolah menggunakan TF-IDF lebih cocok dengan model sederhana, seperti Logistic Regression, dan tidak membutuhkan pemrosesan hubungan antar-node seperti yang digunakan dalam GNN.

Alasan Memilih Logistic Regression

- **Kesederhanaan dan Efektivitas**: Logistic Regression mampu menangani data teks yang diolah secara TF-IDF dengan baik, sehingga memberikan hasil akurasi yang lebih tinggi tanpa kerumitan model neural network.
- **Waktu Komputasi**: Logistic Regression lebih cepat dilatih dan dievaluasi dibandingkan dengan GNN, menjadikannya pilihan lebih praktis untuk data seperti ini.

Rekomendasi untuk Pekerjaan Selanjutnya:

- Meningkatkan keseimbangan data sentimen positif untuk memperbaiki performa model.
- Mengeksplorasi metode GNN lebih lanjut dengan data yang memiliki konektivitas antar entitas.