LAMPIRAN

UNIVERSITAS ISLAM BALITAR
FAKULTAS TEKNOLOGI INFORMASI
Sekretariat / Kampus : Jl. Majapahit No.04 Telp. (0342) 813145 Blitar

Blitar, 21 Februari 2025

Nomor : D.405/SPP/0441/II/2025
Lampiran : -
Perihal : Permohonan Izin Penelitian

Yth. Pengelola Pantai Pudak
Desa Ngadipuro, Kecamatan Wonotirto, Kabupaten Blitar, Provinsi Jawa Timur, Indonesia

Kami beritahukan dengan hormat bahwa mahasiswa tersebut di bawah ini:

Nama	: Muhammad Zidan Asrori Yusuf
NIM	: 20104410070
Program Studi	: Teknik Informatika

Telah kami izinkan untuk menyusun Skripsi guna melengkapi tugas-tugas studi tingkat Sarjana dengan judul penelitian:
"Analisis Sentimen Pengunjung Terhadap Fasilitas Dan Layanan Di Pantai Pudak Menggunakan Algoritma LSTM"

Sehubungan dengan hal tersebut, kami mohon berkenan Saudara untuk menerima mahasiswa kami tersebut untuk melakukan penelitian di institusi dengan waktu pelaksanaan sebagai berikut.

Nama Instansi tujuan	: Pantai Pudak
Alamat	: Desa Ngadipuro, Kecamatan Wonotirto, Kabupaten Blitar, Provinsi Jawa Timur, Indonesia
Waktu	: 17 Januari 2025 - 17 Juli 2025

Atas perhatian dan kerjasama Saudara, kami ucapkan terimakasih.

Dekan Fakultas Teknologi Informatika
Universitas Islam Balitar Blitar

Agus Pangga Kusuma, S.Kom., M.T
NIDN: 0710058506

Surat izin penelitian

Wawancara dengan kepala pengelola

Pemandangan pantai

Warung yang ada di pantai

Toilet pantai

Mushola pantai

Lahan parkir pantai

Akses jalan menuju pantai

Source Code

A. Preprcessing

Import data set yang digunakan

```
import pandas as pd
import numpy as np

data_file = pd.read_excel('dataset.xlsx')
df = pd.DataFrame(data_file[['wiI7pd']])
df.info()
print(df)
```

Cleaning

```
import re

def remove_emoji(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols &
pictographs
```

```
                               u"\U0001F680-\U0001F6FF"  # transport & map
symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               u"\U00002702-\U000027B0"
                               u"\U000024C2-\U0001F251"
                               "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)

def remove_symbols(text):
    text = re.sub(r'[^\w\s]', '', text)
    return text

def remove_numbers(text):
    text = re.sub(r'\d+', '', text)
    return text

df['Cleaning'] = df['wiI7pd'].astype(str).apply(remove_emoji)
df['Cleaning'] = df['wiI7pd'].astype(str).apply(remove_symbols)
df['Cleaning'] = df['wiI7pd'].astype(str).apply(remove_numbers)

df
```

Case folding

```
def case_folding(text):
    text = text.lower()
    return text

df['Casefolding'] = df['Cleaning'].astype(str).apply(case_folding)

df
```

Tokenizing

```
def tokenize(text):
  tokens = text.split()
  return tokens
df['tokenize'] = df['Casefolding'].apply(tokenize)
df.head(10)
```

Stopword removal

```
from nltk.corpus import stopwords
# Tambahkan import nltk di sini
```

```
import nltk

nltk.download('stopwords')
stop_words = stopwords.words('indonesian')

def remove_stopwords(text):
  return [word for word in text if word not in stop_words]

df['stopword_removal'] = df['tokenize'].apply(lambda x:
remove_stopwords(x))
df.head(10)
```

Stemming

```
!pip install Sastrawi

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from nltk.stem import PorterStemmer
from nltk.stem.snowball import SnowballStemmer

factory = StemmerFactory()
stemmer = factory.create_stemmer()

def stem_text(text):
  return [stemmer.stem(word) for word in text]

df['stemming_data'] = df['stopword_removal'].apply(lambda x: '
'.join(stem_text(x)))
df.head(10)
```

B. Labelling Data

Translate stemming

```
# Install library yang dibutuhkan
!pip install pandas
!pip install googletrans==4.0.0-rc1

# Import library
import pandas as pd
from googletrans import Translator
import numpy as np # Import numpy to check for NaN

# Fungsi untuk menerjemahkan teks
```

```
def translate_text(text):
    # Check if the text is not None or NaN before translating
    if text is not None and not pd.isna(text):
        try:
            translator = Translator()
            translated_text = translator.translate(text, src='id',
dest='en')
            return translated_text.text
        except Exception as e:
            # Handle potential translation errors, e.g., print a
warning
            print(f"Error translating text: {text}. Error: {e}")
            return "" # Return empty string or a placeholder on error
    else:
        return "" # Return empty string for None or NaN values

# Baca file excel
df = pd.read_excel('processed_data.xlsx')

# Terjemahkan kolom teks
df['translate_stemming_data'] =
df['stemming_data'].apply(translate_text)
df.head(10)
```

Labelling

```
!pip install nltk
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
import pandas as pd

nltk.download('punkt')
nltk.download('vader_lexicon')
data = SentimentIntensityAnalyzer()
df = pd.read_excel('processed_data+tranlate.xlsx')

labels = []
scores = []

for text in df['translate_stemming_data']:
  if isinstance(text, str):
    sentiment_scores = data.polarity_scores(text)
    compound_scores = sentiment_scores['compound']

    scores.append(compound_scores)

    if compound_scores > 0:
```

```
      label = 'positif'
    elif compound_scores < 0:
      label = 'negatif'
    else:
      label = 'netral'

    labels.append(label)
  else:
    labels.append('netral')
    scores.append(np.nan)

df['compound_score'] = scores
df['sentiment'] = labels

data2 =
df[['wiI7pd','stemming_data','translate_stemming_data','compound_score'
,'sentiment']]
data2.head(10)
```

C. Pembobotan Term Frequency-Inverse Document Drequency (TF-IDF)

```
!pip install pandas scikit-learn

import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np
from tabulate import tabulate

data = pd.read_excel('hasil_labelling.xlsx')
df = data[['stemming_data','compound_score','sentiment']]

# Isi nilai NaN di kolom 'stemming_data' dengan string kosong
df['stemming_data'] = df['stemming_data'].fillna('')

tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(df['stemming_data'])

terms = tfidf_vectorizer.get_feature_names_out()
idf = np.log(tfidf_matrix.shape[0] /
(np.count_nonzero(tfidf_matrix.toarray(), axis=0) + 1))

tfidf_df = pd.DataFrame({'term': terms, 'idf': idf})

for i, doc in enumerate(data['stemming_data']):
  tf = tfidf_matrix[i].toarray().flatten()
```

```
  tfidf_df[f'tf_{i}'] = tf
tfidf_df['tf'] = tfidf_df.iloc[:, 3:].sum(axis=1)
tfidf_df.drop(columns=tfidf_df.columns[3:-1], inplace=True)
# TF-IDF
tfidf_df['tf-idf'] = tfidf_df['tf'] * tfidf_df['idf']
# Print the updated DataFrame
tfidf_df2 = tfidf_df[['term','tf','idf','tf-idf']]
tfidf_df2.head(10)
```

D. Implementasi Algoritma LSTM

```
# --- 1. Install Library ---
!pip install tensorflow scikit-learn pandas seaborn matplotlib openpyxl

# --- 2. Import Library ---
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import class_weight
from sklearn.metrics import confusion_matrix, classification_report,
f1_score, ConfusionMatrixDisplay

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

# --- 3. Load Data ---
df = pd.read_excel('hasil_labelling.xlsx')  # Ubah path jika di lokal

# --- 4. Siapkan Teks dan Label ---
texts = df['stemming_data'].astype(str)
labels = df['sentiment']

# Tokenisasi dan Padding
tokenizer = Tokenizer(num_words=5000, oov_token='<OOV>')
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
padded = pad_sequences(sequences, maxlen=50)

# Label encoding
encoder = LabelEncoder()
encoded_labels = encoder.fit_transform(labels)
```

```
# --- 5. Split Data ---
X_train, X_test, y_train, y_test = train_test_split(
    padded, encoded_labels, test_size=0.2, random_state=42,
stratify=encoded_labels
)

# Cek jumlah data total, train, dan test
print("Total data:", len(padded))
print("Data latih:", len(X_train))
print("Data uji:", len(X_test))

# Cek distribusi label
print("\nDistribusi label di data latih:", np.bincount(y_train))
print("Distribusi label di data uji:", np.bincount(y_test))

# --- 6. Hitung class_weight untuk imbalanced data ---
class_weights = class_weight.compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train),
    y=y_train
)
class_weights_dict = {i : class_weights[i] for i in
range(len(class_weights))}
print("Class weights:", class_weights_dict)

# --- 7. Buat Model LSTM ---
model = Sequential([
    Embedding(input_dim=5000, output_dim=64, input_length=50),
    LSTM(128),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')  # Jumlah kelas: 3 (positif, netral,
negatif)
])

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# --- 8. Training Model ---
model.fit(
    X_train, y_train,
    epochs=20,
    batch_size=32,
    validation_split=0.2,
    class_weight=class_weights_dict
)
```

```
# --- 9.1. Prediksi Data Uji ---
y_pred_test_probs = model.predict(X_test)
y_pred_test = np.argmax(y_pred_test_probs, axis=1)

# Confusion Matrix: Data Uji
cm_test = confusion_matrix(y_test, y_pred_test)
plt.figure(figsize=(6,5))
sns.heatmap(cm_test, annot=True, fmt='d', xticklabels=labels_display,
yticklabels=labels_display, cmap='Blues')
plt.title('Confusion Matrix - Data Uji')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Classification Report: Data Uji
print("\nClassification Report - Data Uji:")
print(classification_report(y_test, y_pred_test,
target_names=labels_display))

# --- 9.2. Prediksi Data Latih ---
y_pred_train_probs = model.predict(X_train)
y_pred_train = np.argmax(y_pred_train_probs, axis=1)

# Confusion Matrix: Data Latih
cm_train = confusion_matrix(y_train, y_pred_train)
plt.figure(figsize=(6,5))
sns.heatmap(cm_train, annot=True, fmt='d', xticklabels=labels_display,
yticklabels=labels_display, cmap='Greens')
plt.title('Confusion Matrix - Data Latih')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()

# Classification Report: Data Latih
print("\nClassification Report - Data Latih:")
print(classification_report(y_train, y_pred_train,
target_names=labels_display))

# --- 9. Evaluasi ---
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
labels_display = encoder.classes_

plt.figure(figsize=(6,5))
```

```
sns.heatmap(cm, annot=True, fmt='d', xticklabels=labels_display,
yticklabels=labels_display, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

# Classification Report
print("Classification Report:\n")
print(classification_report(y_test, y_pred,
target_names=labels_display))

# F1 Macro Score
f1_macro = f1_score(y_test, y_pred, average='macro')
print(f"\nF1 Macro Score: {f1_macro:.4f}")
```

Hasil data yang sudah di olah

https://drive.google.com/drive/folders/1dZpQvOeYZKFNYkk8BTz0Nkq8sIiPdDDv?usp=sharing