**Jurnal Ilmiah Wahana Pendidikan, Mei 2026, 12 (5.A), 128-141**
DOI: https://jurnal.peneliti.net/index.php/JIWP/article/view/14107
p-ISSN: 2622-8327 e-ISSN: 2089-5364
Accredited by Directorate General of Strengthening for Research and Development
Available online at https://jurnal.peneliti.net/index.php/JIWP

# Analisis Sentimen Pengunjung Terhadap Fasilitas Dan Layanan Di Pantai Pudak Menggunakan Algoritma Lstm

**Muhammad Zidan Asrori Yusuf[1], Sri Lestanti[2], Wahyu Dwi Puspitasari[3]**

[1,2,3] Fakultas Teknik Dan Informatika, Universitas Islam Balitar

Received: 6 April 2026
Revised: 6 April 2026
Accepted: 8 April 2026

**Abstract**

*This study aims to analyze visitor sentiment toward the facilities and services at Pudak Beach using the Long Short-Term Memory (LSTM) algorithm. The research adopts an applied study approach with text mining and deep learning-based sentiment analysis techniques. A total of 628 reviews were collected from Google Maps and processed through text preprocessing stages, including cleaning, case folding, tokenizing, stopword removal, and stemming, resulting in 419 clean data ready for analysis. The reviews were then classified into three sentiment categories: positive, neutral, and negative. The classification results showed that 306 reviews (73.03%) were positive, 69 reviews (16.47%) were neutral, and 44 reviews (10.5%) were negative. Model performance was evaluated using a confusion matrix with accuracy, precision, recall, and F1-score metrics. The test results showed that the LSTM model achieved an accuracy of 96% on the training data and 74% on the test data. The highest precision, recall, and F1-score were obtained in the positive sentiment category, while the neutral and negative categories performed lower due to imbalanced data distribution. This study provides a real picture of visitor perceptions and demonstrates the potential of applying big data and Natural Language Processing (NLP) in strategic decision-making in the tourism sector.*

**Keywords:** *Sentiment Analysis, Facilities and Services, Pudak Beach, Long Short-Term Memory, Google Colab*

(*) Corresponding Author: zidanyusuf45@gmail.com,lestanti85@gmail.com, pushpitasari23@gmail.com

**How to Cite:** Yusuf, M., Lestanti, S., & Puspitasari, W. (2026). Analisis Sentimen Pengunjung Terhadap Fasilitas Dan Layanan Di Pantai Pudak Menggunakan Algoritma Lstm. *Jurnal Ilmiah Wahana Pendidikan*, *12*(5.A), 128-141. Retrieved from https://jurnal.peneliti.net/index.php/JIWP/article/view/14107.

## INTRODUCTION

Wisata merupakan aktivitas perjalanan yang dilakukan oleh individu atau kelompok dalam kurun waktu tertentu dengan tujuan rekreasi, pengembangan diri, atau eksplorasi daya tarik suatu destinasi. Objek wisata umumnya ramai dikunjungi wisatawan saat musim liburan untuk menghabiskan waktu bersama orang-orang terdekat, merilekskan pikiran, dan menikmati berbagai aktivitas rekreasi (Khofifah, dkk., 2022). Pantai Pudak, dengan keindahan alamnya yang unik dan memikat, menjadi salah satu destinasi wisata pantai yang banyak dikunjungi di Jawa Timur. Meningkatnya jumlah wisatawan yang berkunjung ke Pantai Pudak menuntut pengelola untuk terus meningkatkan kualitas fasilitas dan layanan yang tersedia.

Peningkatan kualitas fasilitas dan layanan di Pantai Pudak dapat diukur melalui kepuasan pengunjung. Sebanyak 78% wisatawan lebih mengandalkan ulasan online dibandingkan dengan rekomendasi yang diberikan secara langsung oleh individu lain (Ipmawati, dkk., 2024). Dengan demikian, ulasan pengunjung di berbagai platform menjadi aspek yang penting untuk diperhatikan oleh pengelola

tempat wisata. Fasilitas dan layanan yang terdapat pada pantai pudak masih terus dalam tahap pengembangan. Dapat diketahui sebelum pantai pudak menjadi destinasi wisata yang paling diminati oleh pengunjung seperti saat ini, sangat sulit untuk menemukan tempat untuk beribadah maupun toilet. Namun, setelah pantai pudak mengalami lonjakan pengunjung yang melesat naik mau tidak mau pengelola harus berbenah akan fasilitas dan layanan. Seperti halnya akses jalan menuju Pantai Pudak yang dapat dikatakan cukup sulit untuk dilewati yang membuat para pengunjung kesusahan untuk mejangkau Pantai Pudak. Setelah adanya ulasan dari pengunjung pihak pengelola Pantai Pudak mulai berbenah akan akses jalan tersebut. Pihak pengelola memperbaiki akses dengan cara membuat jalan beton sehingga diharapkan para pengunjung dapat mengakses Pantai Pudak dengan mudah.

Kepuasan pengunjung dapat diidentifikasi melalui analisis sentimen terhadap ulasan yang mereka tulis di *platform* daring, seperti *Google Maps*. Dari Ulasan pengunjung di *Google Maps* dapat diketahui sebanyak 628 orang memberikan ulasan yang berbeda-beda dalam memberikan gambaran mengenai persepsi mereka terhadap fasilitas dan layanan yang tersedia di Pantai Pudak. Dalam ulasan tersebut terdapat ulasan positif, netral, maupun negatif. Oleh karena itu, analisis sentimen sangat diperlukan untuk penelitian ini. Analisis sentimen menjadi sangat penting dalam penelitian ini karena berfungsi untuk mengklasifikasikan dan menginterpretasikan opini pengunjung secara otomatis. Dengan mengolah ulasan menggunakan analisis sentimen, dapat diketahui aspek-aspek fasilitas mana yang mendapat respon baik dan mana yang masih perlu diperbaiki. Misalnya, ulasan positif dapat menunjukkan kepuasan terhadap keindahan pantai atau perbaikan akses jalan, sementara ulasan negatif bisa mengungkapkan keluhan tentang kebersihan, ketersediaan toilet, atau fasilitas umum lainnya.

Selain itu, analisis sentimen memungkinkan pengelola Pantai Pudak untuk mengidentifikasi tren dan pola dalam ulasan pengunjung. Dengan memahami sentimen mayoritas, pengelola dapat mengambil keputusan strategis untuk meningkatkan kualitas fasilitas dan layanan, seperti memperbaiki infrastruktur, menambah fasilitas umum, atau meningkatkan kebersihan area wisata. Analisis ini juga membantu dalam memantau efektivitas perubahan yang telah dilakukan, apakah mendapatkan tanggapan positif dari pengunjung atau masih perlu ditingkatkan. Secara keseluruhan, analisis sentimen berperan penting dalam menyediakan wawasan berbasis data yang akurat mengenai persepsi pengunjung. Dengan demikian, pengelola dapat lebih responsif terhadap kebutuhan dan harapan wisatawan, yang pada akhirnya dapat meningkatkan tingkat kepuasan pengunjung dan memperkuat reputasi Pantai Pudak sebagai destinasi wisata unggulan.

Analisis sentimen dapat dilakukan dengan menggunakan algoritma LSTM (*Long Short-Term Memory*). LSTM adalah salah satu jenis *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi keterbatasan RNN dalam menangkap konteks temporal pada data teks. Dengan algoritma ini, analisis sentimen dapat dilakukan dengan mempertimbangkan aspek temporal, sehingga menghasilkan prediksi yang lebih presisi dan mendalam (Putra & Gata, 2024). Algoritma LSTM merupakan salah satu algoritma yang efektif dalam memproses data teks dan memprediksi sentimen. Dengan menggunakan algoritma LSTM, diharapkan dapat diperoleh hasil analisis sentimen yang akurat dan dapat diandalkan.

Hasil analisis sentimen ini dapat menjadi bahan acuan ataupun masukan bagi pengelola untuk meningkatkan kualitas fasilitas dan layanan yang ditawarkan guna memberi kenyamanan pada para pengujung Pantai Pudak. Sehingga dengan adanya penelitian ini diharapkan dapat meningkatkan presentase jumlah pengunjung secara signifikan dan menjadikan Pantai Pudak sebagai destinasi wisata favorit yang lebih baik serta menarik dimata pengunjung.

## METHODS

Penelitian ini berlangsung dari November 2024 hingga Juni 2025, yang terletak di Dusun Banyuurip, Desa Ngadipuro, Kecamatan Wonotirto, Kabupaten Blitar, Jawa Timur dengan fokus pada ulasan pengunjung Pantai Pudak yang dikumpulkan melalui *platform Google Maps*. Penelitian ini menggunakan jenis penelitian terapan yang bertujuan untuk memecahkan masalah praktis atau menghasilkan solusi yang dapat langsung diterapkan untuk menganalisis sentimen pengunjung terhadap fasilitas dan layanan di Pantai Pudak dengan menerapkan algoritma LSTM. Selain teknik scraping, data juga dikumpulkan melalui metode wawancara dan observasi.

## RESULTS & DISCUSSION

### *Results*

1. Hasil Preprocessing Data

Ada 5 tahap penting dalam melakukan *Preprocessing Data* mulai dari *cleaning, case folding, tokenizing, stopword removal dan stemming*.

a. *Cleaning*

*Cleaning* digunakan untuk membersihkan kata yang mengandung *emoji, symbols, serta number.* bertujuan untuk memastikan kualitas dan integritas data yang akan digunakan dalam analisis. Berikut ini adalah *code* nya.

```
import re

def remove_emoji(text):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"
                               u"\U0001F300-\U0001F5FF"
                               u"\U0001F680-\U0001F6FF"
                               u"\U0001F1E0-\U0001F1FF"
                               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
```

*import re* adalah modul untuk operasi dengan ekspresi reguler (*regular expressions*) di *Python*. *Remove_emoji(text)* fungsi ini menggunakan *regular expression (regex)* untuk mengenali rentang karakter *unicode* yang mewakili berbagai emoji dan simbol (seperti emoticon, bendera, ikon transportasi). *Remove_symbol(text)* berfungsi untuk mencari karakter yang bukan huruf dan bukan spasi, seperti tanda baca atau simbol (@#!?$), dan menghapusnya. *Remove_numbers(text)* berfungsi untuk mencari karakter angka dan menghapusnya.

b. *Case Folding*

Pada *Case Folding* data yang awalnya memiliki huruf kapital menjadi huruf kecil semua. Berikut ini merupakan *source code* nya.

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

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

df
```

*Case_folding(text)* berfungsi untuk menerima input text dan mengubah seluruh huruf menjadi huruf kecil menggunakan metode *.lower()*. Fungsi ini bertujuan tahap normalisasi dalam *preprocesing* text agar konsistensi data meningkat dan kata-kata seperti Pantai, pantai, dan PANTAI dianggap sama saat dianalisis.

c. *Tokenizing*

Selanjutnya merupakan tahap *tokenizing,* pada proses *tokenizing* ini data yang sebelumnya berbentuk kalimat akan diubah menjadi perkata. *Source code* nya sebagai berikut.

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

*Tokenize(text)* fungsi ini memecah (membagi) teks menjadi kata-kata (token) berdasarkan spasi. Metode *.split()* secara default akan memecah string berdasarkan spasi, tab, atau *newline*.

d. *Stopword Removal*

*Stopword removal* atau *filtering* digunakau untuk menghapus kata-kata yang di anggap tidak penting. Berikut *source code* nya untuk melakukan *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)
```

*from nltk.corpus import stopwords* baris ini mengimpor modul *stopwords* dari pustaka *NLTK* (*Natural Language Toolkit*), yang berisi daftar stopwords untuk berbagai bahasa. *nltk.download('stopwords')* Baris ini mengunduh daftar *stopwords* dari *server NLTK*. *stop_words = stopwords.words('indonesian')* Baris ini mengambil daftar *stopwords* untuk bahasa Indonesia dan menyimpannya dalam variabel *stop_words*. *def remove_*stopwords*(text):* Fungsi ini menerima teks sebagai input, dan mengembalikan daftar kata-kata yang tidak termasuk dalam daftar stopwords.

e. *Stemming*

*Stemming* merupakan tahap akhir dari proses *pre-processing.* Kuganaan dari *stemming* adalah mengganti kata yang berimbuhan menjadi kata dasar. Berikut adalah *source code* untuk *stemming.*

```
        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)
```

Pertama-tama, dilakukan instalasi pustaka Sastrawi menggunakan *!pip install Sastrawi*. Kemudian, diimpor *StemmerFactory* dari *Sastrawi.Stemmer*, yang digunakan untuk membuat objek stemmer. Meskipun juga diimpor *PorterStemmer* dan *SnowballStemmer* dari pustaka *nltk* (yang sebenarnya digunakan untuk bahasa Inggris), pada bagian kode ini hanya Sastrawi yang digunakan.

Setelah objek stemmer dibuat melalui *factory.create_stemmer()*, selanjutnya didefinisikan sebuah fungsi bernama *stem_text()*. Fungsi ini akan menerima sebuah list kata (hasil dari proses sebelumnya seperti *stopword removal*), lalu menerapkan stemming pada setiap kata dengan menggunakan metode

*stemmer.stem(word).* Hasil dari *stemming* ini kemudian dikembalikan dalam bentuk list kata yang sudah dalam bentuk dasar.

Langkah selanjutnya adalah menerapkan fungsi tersebut ke kolom *'stopword_removal'* dalam DataFrame. Proses ini dilakukan menggunakan *method .apply()* yang akan menerapkan *stem_text()* ke setiap baris. Hasil stemming tersebut kemudian dikembalikan ke bentuk kalimat dengan menggabungkan kata-katanya menggunakan ' '*.join()*, dan disimpan dalam kolom baru bernama '*stemming_data'*.

2. Hasil *Labelling* Data

Setelah melakukan *prepocessing data,* Selanjutnya merupakan tahap *labelling data* untuk mengetahui nilai *compound score* dan termasuk sentimen positif, negatif, atau netral. Sebelum memelakukan *labelling* menggunakan *VADER,* data pada kolom stemming harus di translate dulu agar hasil dari *labelling* lebih akurat. Berikut *source code* nya.

```
!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)
```

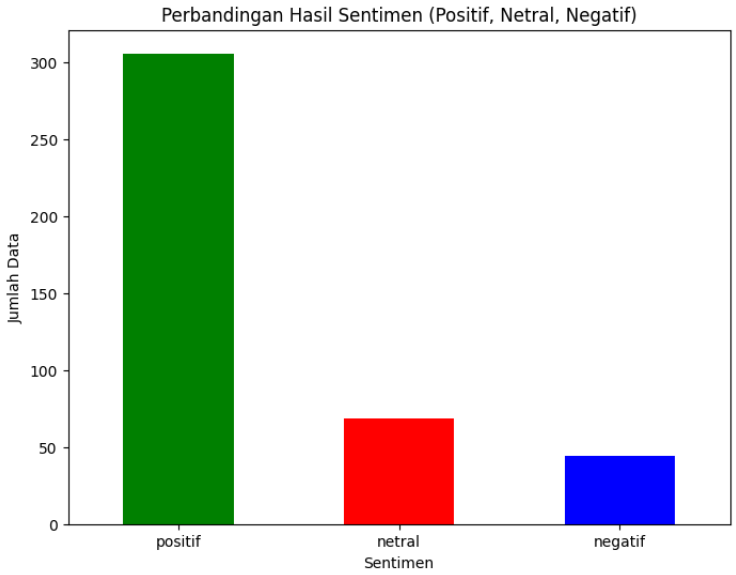
*!pip install googletrans==4.0.0-rc1* digunakan untuk menginstal versi spesifik dari *library Python googletrans*. *from googletrans import Translator* digunakan untuk mengimpor kelas *Translator* dari *library googletrans* ke dalam skrip *Python*.

Setelah proses *translate,* selanjutnya masuk ke proses *labelling*. Berikut ini source code untuk melakukan *labelling* menggunakan *VADER.*

```
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)
```

*data = SentimentIntensityAnalyzer()* digunakan untuk membuat sebuah *instance* dari kelas *SentimentIntensityAnalyzer* dalam pustaka *nltk.sentiment.vader*.

gambar 1 merupakan perbandingan sentimen positif, netral, dan negatif setelah proses *labelling data.*




**Gambar 1** Perbandingan Jumlah Sentimen Positif, Netral, dan Negatif

Gambar 1 merupakan diagram yang menunjukkan perbandingan antara nilai positif 306, nilai netral 69, dan nilai negative 44.

3. *Term Frequency-Inverse Document Drequency* (TF-IDF)

Selanjutnya merupakan pembobotan *Frequency-Inverse Document Drequency* (TF-IDF) pada setiap data. Semakin banyak kata yang berulang pada data, semakin tinggi nilai bobotnya. Berikut ini adalah *source code* untuk pembobotan.

```
!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)
```

*from sklearn.feature_extraction.text import TfidfVectorizer:* Mengimpor kelas *TfidfVectorizer* dari paket *sklearn.feature_extraction.text. TfidfVectorizer* digunakan untuk mengonversi kumpulan dokumen teks menjadi matriks nilai TF-IDF. *from tabulate import tabulate:* Mengimpor fungsi *tabulate* dari paket *tabulate*. *tabulate* digunakan untuk menghasilkan tabel dari data dalam berbagai format. Hasil dari proses pembobotan TF-IDF ditunjukkan pada gambar 4.2.

| | term | tf | idf | tf-idf |
|---|---|---|---|---|
| 0 | abis | 0.473289 | 5.344724 | 2.529601 |
| 1 | ad | 0.218617 | 5.344724 | 1.168445 |
| 2 | ada | 1.378952 | 4.091961 | 5.642619 |
| 3 | adeemm | 0.306798 | 5.344724 | 1.639748 |
| 4 | adrenalin | 0.272253 | 5.344724 | 1.455115 |
| 5 | aduhai | 0.369208 | 5.344724 | 1.973314 |
| 6 | adventure | 0.935035 | 4.939259 | 4.618382 |
| 7 | agam | 0.252253 | 5.344724 | 1.348223 |
| 8 | ahli | 0.590280 | 4.939259 | 2.915544 |
| 9 | air | 3.149528 | 3.398814 | 10.704660 |

**Gambar 2** Hasil Pembobotan TF-IDF

Gambar 2 merupakan hasil pembobotan TF-IDF. Untuk *term* dari data yang di peroleh berjumlah 1049 kata mulai dari abjad A sampai Z.

4. Implementasi *Algoritma Long Short-Term Memory* (LSTM)

Setelah tahap processing data sudah selesai, selanjutnya penerapan Algoritma *Long Short-Term Memory*. Berikut ini adalah *source code* Nya.

```
    !pip install tensorflow scikit-learn pandas seaborn matplotlib
openpyxl
    # --- 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.metrics import confusion_matrix, classification_report,
f1_score, ConfusionMatrixDisplay
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.layers import Embedding, LSTM, Dense,
Dropout
```

*!pip install tensorflow scikit-learn pandas seaborn matplotlib openpyxl, tensorflow* digunakan untuk membuat model *deep learning* (LSTM), *scikit-learn* digunakan untuk preprocessing dan evaluasi model, *pandas, matplotlib, seaborn* digunakan untuk manipulasi data dan visualisasi, *openpyxl* digunakan untuk agar bias membaca/menulis file exel.

```
# --- 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)
```

*x_train, x_test, y_train, y_test* digunakan untuk membagi data uji dan data latih. Dari 419 data diambil 80% sebagai data latih menjadi 335 data dan 20% sebagai data uji menjadi 84 data.

```
# --- 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'])
```

*model = Sequential([...])*model *Sequential* dari *Keras*, yaitu model *linear stack layer* demi *layer*. Model ini cocok digunakan ketika semua lapisan tersusun secara berurutan tanpa percabangan.

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

*model.fit(...)* merupakan fungsi untuk melatih model (*train the model*) menggunakan data pelatihan. Proses ini akan menyesuaikan bobot-bobot di jaringan neural agar model dapat membuat prediksi yang akurat.

```
# 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()
```

*cm = confusion_matrix(y_test, y_pred)* menghasilkan Confusion Matrix dari hasil prediksi *(y_pred)* dibandingkan dengan label sebenarnya *(y_test)*.

```
Classification Report - Data Uji:
              precision    recall  f1-score   support

     negatif       0.30      0.33      0.32         9
      netral       0.56      0.36      0.43        14
     positif       0.83      0.89      0.86        61

    accuracy                           0.74        84
   macro avg       0.56      0.53      0.54        84
weighted avg       0.73      0.74      0.73        84
```

**Gambar 3** Akurasi dan Klasifikasi dari Data Uji

Pada gambar 3 menunjukkan bahwa *Accuracy* model ini mengklasifikasikan 0.74 yang artinya memiliki akurasi 74% data uji dengan benar dari keseluruhan data uji. *Classification report* memberikan informasi lebih mendetail tentang kinerja model untuk setiap kelas.

*Precision* adalah rasio antara prediksi positif yang benar dengan total prediksi positif. Untuk kelas negatif 0.30 di antaranya benar-benar negatif. Untuk kelas netral 0.56 di antaranya benar-benar netral. Untuk kelas positif 0.83 di antaranya benar-benar positif.

*Recall* adalah rasio antara prediksi positif yang benar dengan total contoh positif sebenarnya. Untuk kelas negatif 0.33 dari semua contoh yang sebenarnya negatif. Untuk kelas netral 0.36 dari semua contoh yang sebenarnya netral. Untuk kelas positif 0.89 dari semua contoh yang sebenarnya positif.

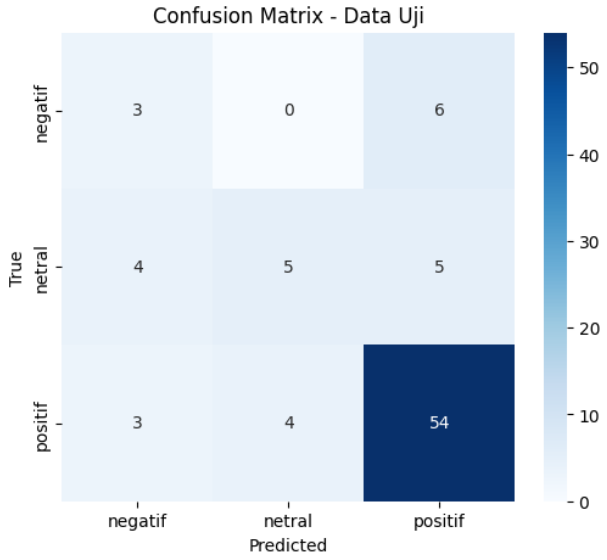
*F1-Score* adalah rata-rata harmonis dari *precision* dan *recall*. Ini memberikan gambaran seimbang dari kedua metrik tersebut. Untuk kelas negatif 0.32, kelas netral 0.43, dan kelas positif 0.86.

*Support* adalah jumlah contoh yang sebenarnya dari masing-masing kelas dalam data uji. Untuk kelas negatif 9 contoh, kelas netral 14 contoh, dan kelas positif 61 contoh.

*Macro avg* menghitung rata-rata dari masing-masing metrik (*precision*, *recall*, *f1-score*) tanpa mempertimbangkan jumlah contoh di setiap kelas. Hasil rata-rata untuk adalah 0.56 untuk *precision*, 0.53 untuk *recall*, dan 0.54 untuk *F1-score.*

*Weighted avg* menghitung rata-rata dari masing-masing metrik (*precision*, *recall, f1-score*) dengan mempertimbangkan jumlah contoh di setiap kelas. Untuk *precision* 0.73, *recall* 0.74, dan *F1-score* 0.73.

Untuk *Confusion matrix* ditunjukkan pada gambar 4 merupakan hasil evaluasi model menggunakan data uji.



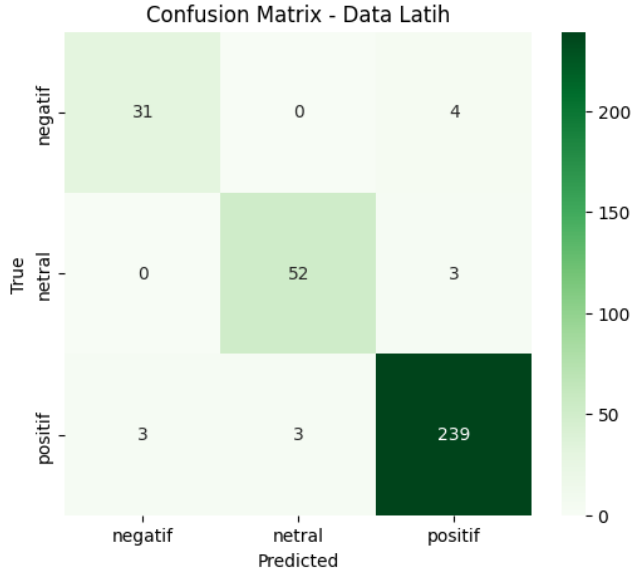

**Gambar 4** *Confusion Matrix* dari Data Uji

Gambar 5 dan gambar 6 merupakan evaluasi model menggunakan data latih.

Classification Report - Data Latih:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| negatif | 0.91 | 0.89 | 0.90 | 35 |
| netral | 0.95 | 0.95 | 0.95 | 55 |
| positif | 0.97 | 0.98 | 0.97 | 245 |
| accuracy | | | 0.96 | 335 |
| macro avg | 0.94 | 0.94 | 0.94 | 335 |
| weighted avg | 0.96 | 0.96 | 0.96 | 335 |

**Gambar 5** Akurasi dan Klasifikasi dari Data Latih




**Gambar 6** *Confusion Matrix* dari Data Latih

## *DISCUSSION*

Penelitian ini bertujuan untuk menganalisis sentimen pengunjung terhadap fasilitas dan layanan di Pantai Pudak dengan menggunakan algoritma *Long Short-Term Memory* (LSTM). Berdasarkan hasil pengujian model, diperoleh bahwa model LSTM mampu mengklasifikasikan ulasan pengunjung ke dalam tiga kategori sentimen, yaitu positif, netral, dan negatif. Mayoritas ulasan menunjukkan sentimen positif, yang menunjukkan bahwa secara umum pengunjung merasa puas terhadap kondisi Pantai Pudak, baik dari segi keindahan alam, kebersihan pantai, maupun adanya fasilitas penunjang seperti tempat makan dan area berkemah.

Hasil ini sesuai dengan teori analisis sentimen yang dijelaskan dalam landasan teori, di mana analisis sentimen bertujuan untuk mengidentifikasi orientasi opini (positif, negatif, netral) dalam teks yang bersifat subjektif (Santoso, dkk., 2017). Dalam konteks penelitian ini, LSTM digunakan sebagai pendekatan untuk melakukan *fined-grained* sentiment analysis karena pemrosesan dilakukan berdasarkan per kalimat dalam ulasan pengunjung. Model LSTM secara efektif menangkap informasi sekuensial dari teks ulasan sehingga dapat mengidentifikasi kata-kata yang bersifat ekspresif dan bermakna secara emosional.

Meski demikian, hasil klasifikasi menunjukkan bahwa model memiliki kelemahan dalam membedakan antara sentimen netral dan negatif. Hal ini tercermin dari hasil *confusion matrix* yang memperlihatkan adanya *false prediction* pada kelas netral dan negatif. Kelemahan ini kemungkinan besar disebabkan oleh ketidakseimbangan data, di mana jumlah ulasan berlabel positif jauh lebih banyak dibandingkan netral dan negatif. Selain itu, secara linguistik, ulasan dalam bahasa Indonesia informal sering menggunakan struktur kalimat yang ambigu, sehingga mempersulit model dalam mengklasifikasikan sentimen secara tepat.

Hasil ini diperkuat oleh kajian literatur dari penelitian Putra & Gata (2024) yang menyebutkan bahwa meskipun LSTM mampu memberikan akurasi tinggi dalam analisis sentimen, kualitas data dan keseimbangan label sangat mempengaruhi kinerja model. Oleh karena itu, dalam pengembangan selanjutnya, perlu diterapkan teknik seperti *word embedding*, *oversampling*, atau *attention mechanism* untuk mengoptimalkan performa klasifikasi, terutama pada data netral dan negatif.

Dengan demikian, penerapan LSTM dalam penelitian ini tidak hanya memberikan gambaran akurat mengenai persepsi pengunjung terhadap fasilitas dan layanan di Pantai Pudak, tetapi juga memperlihatkan potensi besar dari pemanfaatan teknologi *Natural Language Processing* (NLP) dalam mendukung sektor pariwisata melalui pemahaman opini publik secara otomatis dan sistematis

## CONCLUSION

Dari penelitian yang telah dilakukan, Kesimpulan yang dapat di ambil penulis sebagai berikut :

1. Implementasi algoritma LSTM berhasil diterapkan dalam analisis sentimen ulasan pengunjung terhadap fasilitas dan layanan di Pantai Pudak. Proses ini mencakup tahapan *pre-processing* data teks, pelabelan sentimen, dan pemodelan dengan LSTM menggunakan *platform Google Colab*. Hasil dari analisis sentimen menunjukkan bahwa mayoritas ulasan pengunjung memiliki sentimen positif, yaitu sebanyak 306 ulasan (73,03%) dari total

419 data yang dianalisis setelah *preprocessing*. Sementara itu, sentimen netral ditemukan pada 69 ulasan (16,47%) dan sentimen negatif tercatat sebanyak 44 ulasan (10,5%). Ulasan positif mencerminkan apresiasi pengunjung terhadap keindahan alam, kebersihan, dan peningkatan fasilitas seperti akses jalan yang sudah dibeton. Ulasan netral umumnya bersifat informatif tanpa penilaian emosional, dan ulasan negatif banyak mengkritisi keterbatasan infrastruktur serta fasilitas umum yang masih belum memadai.

2. Hasil evaluasi performa model menunjukkan bahwa LSTM mampu mengklasifikasikan sentimen dengan baik, terutama untuk kategori positif. Meskipun terdapat beberapa kekeliruan dalam klasifikasi netral dan negatif, secara umum model memiliki tingkat akurasi yang dapat diterima untuk pengambilan keputusan berbasis data. Didapatkan *accuracy* 74%, untuk kategori positif *precision* 83% *recall* 89% *f1-score* 86%, untuk kategori negatif *precision* 30% *recall* 33% *f1-score* 32% dan untuk kategori netral *precision* 56% *recall* 36% *f1-score* 43%. Hal ini menunjukkan bahwa model mampu mengenali pola kata dan struktur kalimat yang mengandung ekspresi positif secara konsisten. Namun demikian, performa model pada klasifikasi sentimen netral dan negatif masih menunjukkan kelemahan, yang ditandai dengan adanya kesalahan klasifikasi (*false prediction*). Hal ini dapat disebabkan oleh keterbatasan jumlah data netral dan negatif yang tidak seimbang, serta adanya kemiripan penggunaan kata dalam ulasan yang bersifat netral dan negatif. Oleh karena itu, untuk peningkatan performa di masa mendatang, perlu dilakukan penyeimbangan data dan pemanfaatan teknik *embedding* atau *attention mechanism* agar model lebih peka terhadap konteks kalimat.

## REFERENCES

Ipmawati, J., Saifulloh, & Kusnawi. (2024). Analisis Sentimen Tempat Wisata Berdasarkan Ulasan pada Google Maps Menggunakan Algoritma Support Vector Machine. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, *4*(1), 247–256. https://doi.org/10.57152/malcom.v4i1.1066

Khofifah, W., Rahayu, D. N., & Yusuf, A. M. (2022). Analisis Sentimen Menggunakan Naive Bayes Untuk Melihat Review Masyarakat Terhadap Tempat Wisata Pantai Di Kabupaten Karawang Pada Ulasan Google Maps. *Jurnal Interkom: Jurnal Publikasi Ilmiah Bidang Teknologi Informasi Dan Komunikasi*, *16*(4), 28–38. https://doi.org/10.35969/interkom.v16i4.192

Putra, S. A., & Gata, W. (2024). Analisis Sentimen Komentar Youtube Terhadap Ceramah Ning Umi Laila Sindir Rhoma Irama Menggunakan Algoritma LSTM. *Jurnal Ilmiah Komputer*, 692–701.

Santoso, V. I., Virginia, G., & Lukito, Y. (2017). Penerapan Sentiment Analysis Pada Hasil Evaluasi Dosen Dengan Metode Support Vector Machine. *Jurnal Transformatika*, *14*(2), 72. https://doi.org/10.26623/transformatika.v14i2.439