LAMPIRAN

Lampiran 1. *Source Code Program*

```
import os
from bs4 import BeautifulSoup
import requests
import pandas as pd
from tqdm import tqdm

# SCRAP FUNCTION FOR SPECIFIC MONTH
def get_url_list(month=None, year=None):
  base_url =
"https://discover.abc.net.au/index.html?siteTitle=news#/?quer
y=bali%20tourism&page="
  url_list = []

  print(f"Getting url list for month:{month}, year:{year}")
  for date in range(1,32):
    date_url =
f'{base_url}{str(month).zfill(2)}/{str(date).zfill(2)}/{year}
'
    response_page = requests.get(date_url)
    soup = BeautifulSoup(response_page.text, "html.parser")

    for anchor in soup.find_all('a', href=True):
      url = anchor['href']
      if url[:28] == "https://www.abc.net.au/news/":
        if url not in url_list:
          url_list.append(url)

  print(f"Found {len(url_list)} links")
  return url_list

# EXTRACT (SINGLE) PAGE CONTENT FROM GIVEN URL
def scrap_page(url) :

    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")

    title = ""
    for p in soup.find_all('p', {'class':
'paragraph_paragraph___QITb'}):
      title += p.text

    return {"title": title.strip()}
```

```
# EXTRACT (MULTIPLE) PAGE CONTENT FROM URL LIST
def scrap_all(url_list = []):
  scrap_data = []
  for page_url in tqdm(url_list):
    scrap_data.append(scrap_page(page_url))
  return scrap_data
```

L1. *Code Scraping*

```
data = data.reset_index(drop=True)
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))
stemmer = PorterStemmer()

def clean_text(text):
    if not isinstance(text, str):
        return ''
    text = text.lower() # lowercase text
    text = REPLACE_BY_SPACE_RE.sub(' ', text).
    text = BAD_SYMBOLS_RE.sub('', text)
    text = text.replace('x', '')
    # text = re.sub(r'\W+', '', text)
    text = ' '.join(word for word in text.split() if word not
in STOPWORDS) # remove stopwors from text
    return text

def stem_text(text):
    words = text.split()
    stemmed_words = [stemmer.stem(word) for word in words]
    return ' '.join(stemmed_words)

def remove_extra_spaces(text):
   whitespace characters with a single space
    cleaned_text = re.sub(r'\s+', ' ', text)
    return cleaned_text

# Assuming 'data' is a DataFrame with a 'news' column
data = data.reset_index(drop=True)

data['news'] = data['news'].apply(clean_text)
data['news'] = data['news'].apply(stem_text)
data['news'] = data['news'].apply(remove_extra_spaces)
```

L2. *Code* proeses *preprocessing*

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM,
SpatialDropout1D
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.layers import Dropout
import re
from nltk.corpus import stopwords
from nltk import word_tokenize
```

L3. *Code* untuk *Package* dan *Library LSTM*

```
# The maximum number of words to be used. (most frequent)
MAX_NB_WORDS = 86617
# Max number of words in each complaint.
MAX_SEQUENCE_LENGTH = 250
# This is fixed.
EMBEDDING_DIM = 100
tokenizer = Tokenizer(num_words=MAX_NB_WORDS)
tokenizer.fit_on_texts(data['news'].values)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

L4. *Code* untuk Ekstraksi Fitur *Tokenizer*

```
X = tokenizer.texts_to_sequences(data['news'].values)
X = pad_sequences(X, maxlen=MAX_SEQUENCE_LENGTH)
print(X)
```

L5. *Code* untuk Ekstraksi Fitur *Padding*

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,
test_size = 0.10)
```

L6. *Code Splitting* Data

```
model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length
= X.shape[1]))
model.add(SpatialDropout1D(0.8))
model.add(LSTM(100, dropout=0.8, recurrent_dropout=0.8))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

L7. *Code Training* Model

```
earlystop = EarlyStopping(monitor='val_accuracy', patience=3,
verbose=1, mode='max', restore_best_weights=True)

# function for training model and get the best model
val_accuracy
def train_model(model, X_train, Y_train, X_test, Y_test):
    history = model.fit(X_train, Y_train, batch_size=32,
epochs=15, validation_data=(X_test, Y_test), verbose=1,
callbacks=[earlystop])
    # get the best model val_accuracy
    best_val_accuracy = max(history.history['val_accuracy'])
    print('Best val_accuracy: %.2f' %
(best_val_accuracy*100))

    return history

# train model
history = train_model(model, X_train, Y_train, X_test,
Y_test)
```

L8. *Code compile Training* dan *Testing*

```
data2 =
pd.read_csv('/content/drive/MyDrive/Skripsi/data/output_data.
csv')
data2
```

L9. *Code Input* Data

```
tokenizer.fit_on_texts(data2.news)
```

L10. *Code Tokenize* Data

```
def predict_data(model, data2):
    # predict data2
    sequence = tokenizer.texts_to_sequences(data2.news)
    padded = pad_sequences(sequence, maxlen=56,
padding='post', truncating='post')
    pred = model.predict(padded)
    labels = ['Negatif', 'Netral', 'Positif']
    pred_label = [labels[np.argmax(i)] for i in pred]
    data2['Sentiments LSTM'] = pred_label
    return data2

pred = predict_data(model, data2)
pred
```

L11. *Code* Pengujian Data Baru

```
from gensim import corpora
from gensim.models import LdaModel
from gensim.models import CoherenceModel

# Create a dictionary and a corpus
dictionary = corpora.Dictionary(tokenized_documents)
corpus = [dictionary.doc2bow(doc) for doc in
tokenized_documents]

# Initialize lists to store the number of topics and their
corresponding coherence scores
num_topics_list = []
coherence_scores = []

# Define a range of topic numbers to experiment with
topic_range = range(2, 21)  # For example, from 2 to 20
topics

# Loop through different numbers of topics and calculate
coherence scores
for num_topics in topic_range:
    lda_model = LdaModel(corpus=corpus, id2word=dictionary,
num_topics=num_topics, passes=15)
```

```
    coherence_model = CoherenceModel(model=lda_model,
texts=tokenized_documents, dictionary=dictionary,
coherence='c_v')
    coherence_score = coherence_model.get_coherence()

    num_topics_list.append(num_topics)
    coherence_scores.append(coherence_score)

# Plot the coherence scores
plt.figure(figsize=(10, 6))
plt.plot(num_topics_list, coherence_scores, marker='o',
linestyle='-')
plt.title('LDA Model Coherence Scores')
plt.xlabel('Number of Topics')
plt.ylabel('Coherence Score')
plt.grid(True)
plt.show()
```

L12. *Code* Menghitung *Coherence Score* dalam Penentuan Jumlah Topik

```
dictionary = corpora.Dictionary(doc_clean)
print(dictionary)

doc_term_matrix = [dictionary.doc2bow(doc) for doc in
doc_clean]
# Creating the object for LDA model using gensim library
Lda = gensim.models.ldamodel.LdaModel

total_topics = 4 # jumlah topik yang akan di extract
number_words = 10 # jumlah kata per topik

# Running and Trainign LDA model on the document term matrix.
lda_model = Lda(doc_term_matrix, num_topics=total_topics,
id2word = dictionary, passes=50)

lda_model.show_topics(num_topics=total_topics,
num_words=number_words)
```

L13. *Code* Membuat Model *LDA*

```
import pyLDAvis.gensim
import pickle
import pyLDAvis
import os
```

```
LDAvis_data_filepath =
os.path.join('ldavis_prepared_'+str(total_topics))

corpus = [dictionary.doc2bow(text) for text in doc_clean]

if 1 == 1:
    LDAvis_prepared = pyLDAvis.gensim.prepare(lda_model,
corpus, dictionary)
    with open(LDAvis_data_filepath, 'wb') as f:
        pickle.dump(LDAvis_prepared, f)

with open(LDAvis_data_filepath, 'rb') as f:
    LDAvis_prepared = pickle.load(f)

pyLDAvis.save_html(LDAvis_prepared, 'drive/My Drive/Colab
Notebooks/LDA Indonesia/ldavis_prepared_'+ str(total_topics)
+'.html')
LDAvis_prepared
```

L14. *Code* Visualisasi *LDA* dengan *Intertopic Distance Map*

Lampiran 2. Identitas Pembantu Peneliti

DATA PAKAR PELABELAN DATA

Pakar 1

Nama	Ngakan Gde Purnama Putra, S.Pd.
Jenis Kelamin	Laki – Laki
Lulusan	S1 Pendidikan Bahasa Inggris Undiksha 2017
Profesi	Guru Bahasa Inggris

Riwayat Hidup

Ngakan Made Krisna Sedana lahir di Susut Kaja pada tanggal 29 Mei 2001. Penulis lahir dari pasangan suami istri Bapak Nyoman Oka dan Ibu Desak Putu Griastini. Penulis berkebangsaan Indonesia dan beragama Hindu. Kini penulis beralamat di Jl. Merpati No. 3 Gianyar, Kabupaten Gianyar, Provinsi Bali. Penulis menyelesaikan pendidikan dasar di SD N 1 Susut dan lulus pada 2013. Kemudian penulis melanjutkan di SMP Negeri 3 Susut dan lulus pada tahun 2016. Pada tahun 2019, penulis lulus dari SMA N 1 Tampaksiring dengan jurusan IPA. Penulis terdaftar sebagai mahasiswa Program Studi S1 Ilmu Komputer di Universitas Pendidikan Ganesha pada tahun 2019.